Johannes Blömer · Ilias S. Kotsireas
Temur Kutsia · Dimitris E. Simos (Eds.)

LNCS 10693

Mathematical Aspects of Computer and Information Sciences

7th International Conference, MACIS 2017
Vienna, Austria, November 15–17, 2017
Proceedings

Lecture Notes in Computer Science 10693

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

More information about this series at http://www.springer.com/series/7407

Johannes Blömer · Ilias S. Kotsireas
Temur Kutsia · Dimitris E. Simos (Eds.)

Mathematical Aspects of Computer and Information Sciences

7th International Conference, MACIS 2017
Vienna, Austria, November 15–17, 2017
Proceedings

Springer

Editors
Johannes Blömer
University of Paderborn
Paderborn
Germany

Ilias S. Kotsireas
Wilfrid Laurier University
Waterloo, ON
Canada

Temur Kutsia
Johannes Kepler University Linz
Linz
Austria

Dimitris E. Simos
SBA Research
Vienna
Austria

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-72452-2 ISBN 978-3-319-72453-9 (eBook)
https://doi.org/10.1007/978-3-319-72453-9

Library of Congress Control Number: 2017961797

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Mathematical Aspects of Computer and Information Sciences (MACIS) is a series of biennial conferences focusing on research in mathematical and computational aspects of computing and information science. It is broadly concerned with algorithms, their complexity, and their embedding in larger logical systems. At the algorithmic level there is a rich interplay along the numerical/algebraic/geometrical/topological axes. At the logical level, there are issues of data organization, interpretation, and associated tools. These issues often arise in scientific and engineering computation where we need experiments and case studies to validate or enrich the theory. At the application level, there are significant applications in the areas of mathematical cryptography, machine learning, and data analysis, and the various combinatorial structures and coding theory concepts that are used in a pivotal role in computing and information sciences. MACIS is interested in outstanding and emerging problems in all these areas. Previous MACIS conferences have been held in Beijing (2006, 2011), Paris (2007), Fukuoka (2009), Nanning (2013), and Berlin (2015). MACIS 2017 was held at the University of Applied Sciences Technikum Wien located in the capital of Austria.

We are grateful to the track chairs and the Program Committee for their critical role in putting together a very successful technical program, especially under strict deadlines. We also wish to extend our gratitude to all MACIS 2017 conference participants – all of them contributed to making the conference a success. The conference would not have been possible without the hard work of the local organizer from SBA Research, Bettina Bauer. SBA Research is Austria's leading center dedicated to information security. One of its core research areas focuses on mathematical aspects of information security. We are extremely fortunate to have received the generous support of our sponsors: University of Applied Sciences Technikum Wien, Zuse Institute Berlin (in particular, we are thankful to Winfried Neun for his efforts toward transferring the past MACIS budget to this edition of MACIS), SBA Research, and the Vienna Convention Bureau. Last but not least, we are thankful to the two invited speakers, Bruno Buchberger (RISC, Johannes Kepler University, Austria) and Dongming Wang (Beihang University, China and CNRS, France), for honoring the conference with their participation and stimulating talks.

The volume contains 36 refereed papers (28 regular and 8 short papers) carefully selected out of 67 total submissions (53 regular, 14 short); thus, MACIS 2017 has an overall acceptance rate of 54%. The papers are organized in different categories corresponding to four tracks featured in the MACIS 2017 conference. The topics of the MACIS 2017 tracks cover a wide array of research areas, as follows:

Track 1: Foundation of Algorithms in Mathematics, Engineering and Scientific Computation
Track Chairs: Matthew England, Jonathan Hauenstein, Laura Kovács, Elias Tsigaridas

Track 2: Combinatorics and Codes in Computer Science
Track Chairs: Daniel Augot, Alexander May, Alfred Wasserman
Track 3: Data Modeling and Analysis
Track Chairs: Xiaoyu Chen, Joachim Giesen, Giorgos Kollias
Track 4: Mathematical Aspects of Information Security and Cryptography
Track Chairs: Jan Camenisch, Stefan Dziembowski, Guenael Renault

We wish to thank all the track chairs for their hard work in putting together these tracks. Last but not least, we thank the Springer management and production team for their support.

October 2017

Johannes Blömer
Ilias S. Kotsireas
Temur Kutsia
Dimitris E. Simos

Organization

General Chair

Dimitris E. Simos	SBA Research, Austria

Local Organization

Bettina Bauer	SBA Research, Austria
Yvonne Poul	SBA Research, Austria

Program Chairs

Johannes Blömer	Paderborn University, Germany
Temur Kutsia	RISC, Johannes Kepler University, Austria

Program Committee

Daniel Augot	Inria, France
Dan Bates	Colorado State University, USA
Russell Bradford	University of Bath, UK
Daniel Brake	University of Wisconsin Eau Claire, USA
Laurent Buse	Inria, France
Eimear Byrne	University College Dublin, Ireland
Jan Camenisch	IBM Research, Switzerland
Xiaoyu Chen	Beihang University, China
Stefan Dziembowski	University of Warsaw, Poland
Matthew England	Coventry University, UK
Joachim Giesen	Friedrich-Schiller-Universität Jena, Germany
Alberto Griggio	FBK-IRST, Italy
Jonathan Hauenstein	University of Notre Dame, USA
Giorgos Kollias	IBM Research USA
Ilias Kotsireas	Wilfrid Laurier University, Canada
Laura Kovacs	Vienna University of Technology, Austria
Alexander May	Ruhr-Universität Bochum, Germany
Winfried Neun	ZIB, Germany
Rodrigue Ngueyep	IBM Research, USA
Athanasios Nikolakopoulos	University of Patras, Greece
Veronika Pillwein	Johannes Kepler University, Austria
Guenael Renault	ANSSI/UPMC LIP6/Inria POLSYS, France
Elias Tsigaridas	Inria, France
Alfred Wassermann	Universität Bayreuth, Germany

Additional Reviewers

Bachoc, Christine
Both, Leif
Chen, Changbo
Esser, Andre
Gaborit, Philippe
Guibert, Olivier
Guillevic, Aurore
Heuer, Felix
Irfan, Ahmed
Kampel, Ludwig
Kremer, Gereon
Kübler, Robert
Lavauzelle, Julien
Levandovskyy, Viktor
Magliveras, Spyros
Matsuda, Takahiro
Ostergard, Patric
Pavcevic, Mario Osvin
Robertz, Daniel
Seiler, Gregor
Seurin, Yannick
Tillich, Jean-Pierre
Tonchev, Vladimir

MACIS Steering Committee

Ilias Kotsireas	Wilfrid Laurier University, Canada (Chair)
Siegfried Rump	Hamburg University of Technology, Germany
Thomas Sturm	CNRS, France
Dongming Wang	Beihang University, China and CNRS, France
Jinzhao Wu	Guangxi University for Nationalities, China
Chee Yap	New York University, USA

Contents

Foundation of Algorithms in Mathematics, Engineering and Scientific Computation

Foundation of Algorithms in Mathematics, Engineering and Scientific Computation

Automated Reasoning for Knot Semigroups and π-orbifold Groups of Knots

Alexei Lisitsa[1(✉)] and Alexei Vernitski[2(✉)]

[1] Department of Computer Science, University of Liverpool, Liverpool, UK
A.Lisitsa@liverpool.ac.uk
[2] Department of Mathematical Sciences, University of Essex, Essex, UK
asvern@essex.ac.uk

Abstract. The paper continues the first author's research which shows that automatic reasoning is an effective tool for establishing properties of algebraic constructions associated with knot diagrams. Previous research considered involutory quandles (also known as keis) and quandles. This paper applies automated reasoning to knot semigroups, recently introduced and studied by the second author, and π-orbifold groups of knots. We test two conjectures concerning knot semigroups (specifically, conjectures aiming to describe knot semigroups of diagrams of the trivial knot and knot semigroups of 4-plat knot diagrams) on a large number of examples. These experiments enable us to formulate one new conjecture. We discuss applications of our results to a classical problem of the knot theory, determining whether a knot diagram represents the trivial knot.

1 Main Definitions

Knot theory is an important part of topology because knots are, in a sense, simplest three-dimensional objects. Studying two-dimensional knot diagrams and studying algebraic constructions arising from knots are two of the most important techniques of knot theory [17]. Frequently (as in this paper) these two approaches are combined. This paper uses automated reasoning to improve our understanding of some known and some new algebraic constructions related to knots and knot diagrams.

1.1 Arcs and Crossings

By an *arc* we mean a continuous line on a knot diagram from one undercrossing to another undercrossing. For example, consider the knot diagram $\mathfrak{t}$ on Fig. 1;

Alexei Lisitsa—Part of this research was carried out during visits by the first named author to the Department of Mathematical Sciences at the University of Essex in 2016. The visits were financed by a London Mathematical Society Scheme 7 Grant (ref. SC7-1516-12). This research has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. H2020-FETOPEN-2015-CSA 712689 (SC^2).

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 3–18, 2017.
https://doi.org/10.1007/978-3-319-72453-9_1

it has three arcs, denoted by a, b and c. To denote a crossing on a knot diagram we shall use notation $x \dashv y \vdash z$, where x and z are the two arcs terminating at the crossing and y is the arc passing over the crossing. For example, the crossings on diagram $\mathfrak{t}$ are $b \dashv a \vdash c$, $b \dashv a \vdash a$ and $c \dashv a \vdash a$.

1.2 Cancellative Semigroups and Knot Semigroups

A semigroup is called *cancellative* if it satisfies two conditions:

$$\text{if } xz = yz \text{ then } x = y, \text{ and if } xy = xz \text{ then } y = z.$$

For each given knot diagram $\mathfrak{d}$, we define a cancellative semigroup which we call the *knot semigroup* of $\mathfrak{d}$ and denote by $K\mathfrak{d}$; the construction has been introduced and studied in [25]. To define the knot semigroup of a diagram $\mathfrak{d}$, assume that each arc is denoted by a letter. Then at every crossing $x \dashv y \vdash z$, 'read' two defining relations

$$xy = yz \text{ and } yx = zy.$$

The cancellative semigroup generated by the arc letters with these defining relations is the knot semigroup $K\mathfrak{d}$ of $\mathfrak{d}$. For example, on diagram $\mathfrak{t}$ we can read relations $ba = ac$ and $ab = ca$ at the left-top crossing, relations $ba = aa$ and $ab = aa$ at the right-top crossing and relations $ca = aa$ and $ac = aa$ at the bottom crossing. Using these relations, one can deduce equalities of words in $K\mathfrak{t}$. In particular, from $aa = ba = ca$, using cancellation, one can deduce $a = b = c$, that is, all generators are equal to one another; in other words, $K\mathfrak{t}$ is an infinite cyclic semigroup.

1.3 Keis

A *kei* (also known as an *involutory quandle*) is defined as an algebra with one binary operation $\triangleright$ and three axioms

$$a \triangleright a = a, \ (a \triangleright b) \triangleright b = a \text{ and } (a \triangleright b) \triangleright c = (a \triangleright c) \triangleright (b \triangleright c).$$

It is useful to know that every group can be considered as a kei with the operation $g \triangleright h = hg^{-1}h$. For a given knot diagram $\mathfrak{d}$, the kei $IQ\mathfrak{d}$ of the knot is a kei generated by the arc letters with defining relations $x \triangleright y = z$ and $z \triangleright y = x$ for each crossing $x \dashv y \vdash z$ of $\mathfrak{d}$. The mnemonic behind notation $x \triangleright y = z$ is expressed in [11]: 'x under y gives z'. The three axioms of a kei directly correspond to the three Reidemeister moves [5].

1.4 π-orbifold Groups and Two-fold Groups

For a given knot diagram $\mathfrak{d}$, the π-orbifold group $O\mathfrak{d}$ of the knot is a group generated by the arc letters with the following relations. For each arc x of the diagram $\mathfrak{d}$, introduce a relation $x^2 = 1$. At every crossing $x \dashv y \vdash z$, introduce a defining relation $xy = yz$ (or, equivalently, $yx = zy$, or $yxy = z$, or $yzy = x$).

Obviously, $O\mathfrak{d}$ is a factor group of $K\mathfrak{d}$. Denote the generating set of $O\mathfrak{d}$, that is, the set of arcs of $\mathfrak{d}$, by A, and consider the natural homomorphism from the free semigroup A^+ onto $O\mathfrak{d}$. It is easy to see that for each element g of $O\mathfrak{d}$, either only words of an odd length are mapped to g or only words of an odd length are mapped to g. Accordingly, let us say that g is an element of an odd (even) length in the former (latter) case. A subgroup of $O\mathfrak{d}$ consisting of elements of an even length is called the fundamental group of the 2-fold branched cyclic cover space of a knot [20,26]; we shall shorten this name to the *two-fold group of a knot*, and shall denote the group by $T\mathfrak{d}$.

1.5 Putting These Constructions Together

Consider a simple example. The π-orbifold group $O\mathfrak{t}_3$ of the trefoil knot diagram $\mathfrak{t}_3$ is the dihedral group D_3. The group D_3 naturally splits into two types of elements: 3 rotations and 3 reflections. The rotations form a subgroup of D_3, which is the group $T\mathfrak{t}_3$, and which happens to be isomorphic to $\mathbb{Z}_3$. The reflections are in a one-to-one correspondence with the arcs of the trefoil knot diagram, and the subkei of D_3 consisting of reflections is $IQ\mathfrak{t}_3$. Generalising this example, one can notice that every group $O\mathfrak{d}$ splits into the subgroup $T\mathfrak{d}$ consisting of elements of an even length and a subkei consisting of elements of an odd length; this subkei is related to (and in many natural examples is isomorphic to) the kei of the knot $IQ\mathfrak{d}$.

1.6 Other Constructions: Knot Groups and Quandles

If one considers the diagram of a knot as an oriented curve, that is, in the context of a specific prescribed direction of travel along the curve, another pair of algebraic constructions can be introduced (whose definitions we shall skip, because they are not directly related to the topic of the paper). One of them is the *knot group*, which is historically the first and the best known construction (see, for example, Sect. 6.11 in [9] or Chap. 11 in [16]). The other is the *quandle* (also known as a *distributive groupoid*) of a knot; see, for example, [18]. These two constructions are 'larger' than the ones we consider in the sense that the π-orbifold group is a factor group of the knot group, and the kei is a factor kei of the quandle.

2 Trivial Knots

Trivial knots can be characterised via algebaic constructions associated with them, as the following results show.

Fact 1. *The following are equivalent:*

- *A knot is trivial.*
- *The two-fold group of the knot is trivial [20, 26].*
- *The kei of the knot is trivial [26].*

Fig. 1. Knot diagrams $\mathfrak{t}$ and $\mathfrak{t}_3$

– *The group of the knot is trivial [3].*
– *The quandle of the knot is trivial [11].*

Since the π-orbifold group of a knot is 'sandwiched' between the two-fold group of the knot and the group of the knot, the result also holds for π-orbifold groups.

The unusually simple structure of $K\mathfrak{t}$ in the example in Sect. 1 may be related to the fact that $\mathfrak{t}$ is a diagram of the trivial knot: it is easy to see that $\mathfrak{t}$ is not really 'knotted'. A general conjecture was formulated in [25]:

Conjecture 1. A knot diagram $\mathfrak{d}$ is a diagram of the trivial knot if and only if $K\mathfrak{d}$ is an infinite cyclic semigroup.

When Conjecture 1 is fully proved, it will be a natural addition to the list of results in Fact 1. In this paper we test Conjecture 1 on a series of knot diagrams and check how efficient the technique suggested by it is at detecting trivial knots. We conduct three types of computational experiments related to Conjecture 1:

– There are well-known examples of complicated diagrams of the trivial knot. Given one of these diagrams $\mathfrak{d}$, we prove that $K\mathfrak{d}$ is cyclic.
– We consider a number of standard diagrams of non-trivial knots. For each of these diagrams $\mathfrak{d}$, we prove that $K\mathfrak{d}$ is not cyclic or, equivalently, that $O\mathfrak{d}$ is not trivial.
– We can construct complicated diagrams of the trefoil, the simplest non-trivial knot, by considering the sum of the standard trefoil diagram $\mathfrak{t}_3$ and of one the complicated diagrams of the trivial knot. Given a complicated diagram $\mathfrak{d}$ of the trefoil, we check how efficiently automated reasoning proves that $K\mathfrak{d}$ is not cyclic.

For comparison, in [6] using automated reasoning was proposed for unknot detection and experiments with proving and disproving triviality of $IQ\mathfrak{d}$ were conducted. Yet another technique was used in [7]: the problem of checking if a knot is trivial was reduced to comparing factor quandles of the knot with families of pre-computed quandles, and this procedure, in its turn, was reduced to SAT solving.

2.1 How to Test if a Knot Semigroup Is Cyclic

Consider a knot diagram $\mathfrak{d}$ with n arcs $a_1, \ldots, a_n$. Let $R_{\mathfrak{d}}$ be the set of relations read on the crossings of $\mathfrak{d}$, as defined in Sect. 1.

The equational theory of the knot semigroup $K\mathfrak{d}$ is $E_{K\mathfrak{d}} = E_{cs} \cup R_{\mathfrak{d}}$, where E_{cs} is the set of equational axioms of cancellative semigroups (see them listed explicitly in Subsect. 2.2).

By Birkhoff's completeness theorem, two words are equal if and only if their equality can be proved by equational reasoning [1,10]; hence the following statement follows (a similar statement for keis is formulated as Proposition 1 in [6]).

Proposition 1. *A knot semigroup $K\mathfrak{d}$ of a diagram $\mathfrak{d}$ with n arcs $a_1, \ldots, a_n$ is cyclic if and only if $E_{K\mathfrak{d}} \vdash \wedge_{i=1\ldots n-1}(a_i = a_{i+1})$, where $\vdash$ denotes derivability in the equational logic, or, equivalently in the first-order logic with equality.*

Proposition 1 suggests a practical way for experimental testing of Conjecture 1. Given a knot diagram $\mathfrak{d}$ (represented, for example, by its Gauss code [21]), translate it into knot semigroup presentation $R_{\mathfrak{d}}$ and further into its equational theory $E_{K\mathfrak{d}}$. Then apply an automated theorem prover and disprover to the problem $E_{K\mathfrak{d}} \vdash \wedge_{i=1\ldots n-1}(a_i = a_{i+1})$.

Note that if a *complete* prover is used (that is, given a valid formula it eventually produces a proof), the described procedure constitutes a semi-decision algorithm: if a knot semigroup is cyclic then this fact will be eventually established. Most common procedure for disproving is a finite model building [4], which, given a formula, builds a finite model for the formula's negation, thereby refuting the original formula. Usually it is possible to ensure *finite completeness* of a model builder: given a formula, it eventually produces a finite model refuting it, providing such a model exists. In general, however, due to undecidability of first-order logic, no complete disproving procedure is available. In particular, sometimes only infinite models refuting an invalid formula exist; then the model builder cannot build a model, even it is a complete finite model builder.

2.2 Cyclic Knot Semigroups

We applied automated theorem prover Prover9[1] [19] to several well-known diagrams of the trivial knot, and it has successfully proved that the knot semigroup is cyclic in each case. To illustrate the approach we present the proof for the simple diagram $\mathfrak{t}$ in Sect. 1. The task specification for Prover9 is divided into assumptions and goals parts. The assumptions part includes cancellative semigroups axioms E_{cs}:

```
(x * y) * z = x * (y * z).
x * y = x * z -> y = z.
y * x = z * x -> y = z.
```

and defining relations for diagram $\mathfrak{t}$:

[1] We have chosen Prover9 and model builder Mace4 (below), primarily to be able to compare efficiency of automated reasoning with semigroups with that for involutory quandles in [6], where the same systems were used. Otherwise the choice is not very essential and any other automated first order (dis)provers could be used instead.

```
a * a = a * c. c * a = a * b.
b * a = a * a. a * a = c * a.
a * c = b * a. a * b = a * a.
```

The goals part is

```
(a = b) & (b = c).
```

For this task Prover9 produces the proof of length 14 in 0.05 s. Table 1 presents the results for several well-known diagrams of the trivial knot. Time for the proof search grows with the size of the diagram. The diagram Ochiai, II (45 crossings) is a distinctive outlier: for some reason, the proof search for it took more than 8000 s, comparing with 368 s for Haken Gordian diagram with 141 crossings. We do not understand the reasons of why Ochiai, II diagram is so difficult for the automated proof. We are planning to explore this case further and to apply various automated provers and strategies to it.

Table 1. Proving that semigroups of diagrams of the trivial knot are cyclic

Name of unknot	Reference	# of crossings	Time, s
Culprit	[13]	10	0.4
Goerlitz	[12]	11	2.5
Thistlethwaite	[24]	15	6.1
Ochiai, I	[22]	16	14.85
Freedman	[23]	32	38.2
Ochiai, II	[22]	45	8458.6
Ochiai, III	[22]	55	195.2
Haken Gordian	[15]	141	368

2.3 Non-cyclic Knot Semigroups: Small Knots

We applied an automated model builder Mace4 [19] to all standard knot diagrams with up to 9 crossings (a table defining these knots can be found, for example, as Appendix 1 of [17]). The word *model* in this case means a finite non-cyclic factor semigroup of the diagram's knot semigroup. It is useful to note that since every finite cancellative semigroup is a group, Mace4 actually finds a group model. We illustrate the approach by considering the simplest untrivial knot, the trefoil knot (diagram $\mathfrak{t}_3$ in Sect. 1, entry 3_1 in Table 2). The task specification for Mace4 includes the cancellative semigroup axioms (as in Subsect. 2.2) and the defining relations for the knot semigroup of the trefoil knot:

```
a * b = b * c. b * c = c * a. c * a = a * b.
b * a = c * b. c * b = a * c. a * c = b * a.
```

The goal to disprove is

```
(a = b) & (b = c).
```

Mace4 disproves the goal by finding a model in which both the cancellative semigroup axioms and the defining relations are satisfied, but at the same time, the goal statement is false. The model found by Mace4 is the dihedral group D_3, which is the knot's π-orbifold group, as discussed in Subsect. 1.5.

Table 2 shows the results for all standard knot diagrams with up to 9 crossings. For each diagram we list the size of the model found and the time taken to find this model. The results presented in non-bold font are obtained by running Mace4 with the default iterative search strategy; that is, the search for a model starts with the size 2; if no model is found by an exhaustive search of models of a certain size, the size is increased by 1 and the search continues. Thus, assuming correctness of Mace4, entries in non-bold font represent smallest possible models. In all these cases the size of the model is two times the size of a smallest kei model computed in [6]; this observation has led us to formulating the following conjecture.

Conjecture 2. Consider a knot diagram $\mathfrak{d}$. Suppose the kei of $\mathfrak{d}$ has a factor kei of size n. Then the semigroup $K\mathfrak{d}$ has a factor semigroup of size $2n$.

To add some more details regarding the conjecture, the smallest semigroup model is frequently the knot's π-orbifold group, which is frequently (see Fact 2) a dihedral group, and the size of a dihedral group is two times the size of the corresponding *dihedral kei* (that is, the kei consisting of reflections), which is then the smallest kei model of the same knot diagram (Proposition 2 in [6], Theorem 3 in [7]). In some other cases (for example, 8_{19} in Table 2, which is not a 4-plat), the knot's π-orbifold group is not a dihedral group, but the smallest semigroup model, which is is a factor group of the knot's π-orbifold group, happens to be isomorphic to the dihedral group D_3. We don't know what happens to smallest model sizes when the smallest semigroup model is not a dihedral group and the smallest kei model is not a dihedral kei.

Table 2 contains remarks related to Conjecture 2. The entries in bold font represent the diagrams for which the default iterative strategy of Mace4 has failed to find a model in 50000 s. In this case we used Conjecture 2 to guess a possible model size. These entries further split into three categories:

(1) the size is given with a mark^a, meaning the search has been completed successfully for this particular size, predicted by Conjecture 2; the conjecture is confirmed, but the model found is not necessarily minimal;
(2) the size is given with a mark^b, meaning the search has been done for increasing model sizes and ended successfully, but this was not an exhaustive search, as a time limit was imposed on search for each size; Conjecture 2 is confirmed, but the model found is not necessarily minimal;
(3) the size is given with a question mark and the time is given as N/F for 'not found', meaning neither search strategy has succeeded to find a model in 50000 s; an estimated model size is given as predicted by Conjecture 2.

It is interesting to note that we could not find a model of the predicted size 30 in any entry in the table, as Mace4 search has timed out, although for larger

values up to 46, Mace4 was able to find a model of predicted size. It might be just a coincidence, but 30 is the only value in the table which is not two multiplied by a prime number.

Table 2. Models for the standard knot diagrams with at most 9 crossings.

Knot	3_1	4_1	5_1	5_2	6_1	6_2	6_3	7_1
Size	6	10	10	14	6	22	26	14
Time	0.01	0.45	0.30	3.54	0.06	297	1362	5.02
Knot	7_2	7_3	7_4	7_5	7_6	7_7	8_1	8_2
Size	22	26	6	34	**38**^{a}	6	26	**34**^{a}
Time	339	1378	0.05	20193	**5715**	0.08	1484	**285**
Knot	8_3	8_4	8_5	8_6	8_7	8_8	8_9	8_{10}
Size	**34**^{a}	**38**^{a}	6	**46**^{a}	**46**^{a}	10	10	6
Time	**247**	**1350**	0.05	**2569**	**2684**	0.53	1.15	0.08
Knot	8_{11}	8_{12}	8_{13}	8_{14}	8_{15}	8_{16}	8_{17}	8_{18}
Size	6	**58?**	**58?**	**62?**	6	10	**74?**	6
Time	0.12	**N/F**	**N/F**	**N/F**	0.08	2.71	**N/F**	0.09
Knot	8_{19}	8_{20}	8_{21}	9_1	9_2	9_3	9_4	9_5
Size	6	6	6	6	6	**38?**	6	**46**^{b}
Time	0.06	0.06	0.05	0.28	0.22	**N/F**	0.20	**20316**
Knot	9_6	9_7	9_8	9_9	9_{10}	9_{11}	9_{12}	9_{13}
Size	6	**58?**	**62?**	**62?**	6	6	10	**74?**
Time	0.34	**N/F**	**N/F**	**N/F**	0.39	0.19	13.10	**N/F**
Knot	9_{14}	9_{15}	9_{16}	9_{17}	9_{18}	9_{19}	9_{20}	9_{21}
Size	**74?**	6	6	6	**82?**	**82?**	**82?**	**86?**
Time	**N/F**	0.05	0.33	0.16	**N/F**	**N/F**	**N/F**	**N/F**
Knot	9_{22}	9_{23}	9_{24}	9_{25}	9_{26}	9_{27}	9_{28}	9_{29}
Size	**30?**	6	6	**30?**	**94?**	14	6	6
Time	**N/F**	0.09	0.09	**N/F**	**N/F**	65	0.05	0.12
Knot	9_{30}	9_{31}	9_{32}	9_{33}	9_{34}	9_{35}	9_{36}	9_{37}
Size	**30?**	10	**118?**	**122?**	6	6	**30?**	6
Time	**N/F**	2.51	**N/F**	**N/F**	0.09	0.28	**N/F**	0.17
Knot	9_{38}	9_{39}	9_{40}	9_{41}	9_{42}	9_{43}	9_{44}	9_{45}
Size	6	10	6	14	14	**26**^{a}	**30?**	**30?**
Time	0.20	6.35	0.20	117	50.22	**365**	**N/F**	**N/F**
Knot	9_{46}	9_{47}	9_{48}	9_{49}				
Size	6	6	6	10				
Time	0.37	0.09	0.05	9.47				

2.4 Non-cyclic Knot Semigroups: Sums of Knots

We applied automated finite model builder Mace4 to the sums of all named trivial knot diagrams from Table 1 with the trefoil diagram in order to test whether a suitable model can be found by automated reasoning. When applied to the largest Haken Gordian diagram Mace4 generated an error message[2]. For all other sample diagrams the model of the expected size 6 was found (the same model as in the example presented in Subsect. 2.3). The iterative search starting with models of size 2 did not always work because larger diagrams timed out at a 1500 s limit. However, the search through models of size 6 has found a model in under 0.2 s in all cases. The results are shown in Table 3.

Table 3. Search for models for the sum of a trivial knot and the trefoil

Name of unknot	# of crossings in the sum	Started with size 6 Time, s	Started with size 2 Time, s
Culprit	13	0.09	0.45
Goerlitz	14	0.06	1.39
Thistlethwaite	18	0.03	>1500
Ochiai, I	19	0.08	3.92
Freedman	35	0.09	>1500
Ochiai, II	48	0.14	>1500
Ochiai, III	58	0.12	>1500
Haken Gordian	144	N/A	N/A

2.5 Efficiency Comparison

Our experiments demonstrate that automated reasoning using knot semigroups can be applied for unknot detection (providing that Conjecture 1 is true), but it is not as efficient as automated reasoning using keis or quandles.

As to recognising a diagram of the trivial knot, automated reasoning on keis does it in under 1 s for all diagrams in Table 1 (reported in [6]), except Ochiai's unknots and Haken unknot. The sharpest difference is the Ochiai, II diagram, whose kei is proved to be trivial by Prover9 in under 4 s, as compared with more than 8000 s for semigroups. As to Haken unknot, the kei is proved to be trivial in about 15 s, as compared with 368 s for semigroups.

As to finding models for non-trivial knots, Mace4 using knot semigroups (or π-orbifold groups) has reported time out (50000 s) on 23 out of 84 diagrams in Table 2. The corresponding kei models were found in [6] for all 9-crossing diagrams with the average time 28.6 s.

An even more efficient automated reasoning procedure for detecting trivial knots has been obtained in [7] by considering quandles and reducing the problem

[2] `Fatal Error: mace4: domain_element too big.`

of finding a finite factor quandle by to SAT. One reason why detecting trivial knots with keis and quandles is more efficient in practice than with semigroups (or π-orbifold groups) is because in many natural examples, the smallest factor kei of the knot kei is two times smaller than the smallest factor group of the knot semigroup, as discussed in Conjecture 2.

3 4-plats

A 4-plat knot diagram is a braid with 4 strands whose ends on the left-hand side and the right-hand side are connected to form one closed curve, as in the example shown on Fig. 2 (taken from [25]). 4-plat knots, that is, knots represented by 4-plat diagrams, form an important class of knots and are also known as 2-bridge knots and rational knots. Now we shall introduce some concepts which we need to formulate Conjecture 3 which aims to describe knot semigroups of 4-plat diagrams.

Fig. 2. A 4-plat and labelling its arcs

Let $B \subseteq \mathbb{Z}_n$ for some fixed positive integer n. By the alternating sum of a word $b_1b_2b_3b_4 \dots b_k \in B^+$ we shall mean the value of the expression $b_1 - b_2 + b_3 - b_4 + \dots + (-1)^{k+1}b_k$ calculated in $\mathbb{Z}_n$. We shall say that two words $u, v \in B^+$ are in relation $\sim$ if and only u and v have the same length and the same alternating sum. It is obvious that $\sim$ is a congruence on B^+. We denote the factor semigroup $B^+/\sim$ by $AS(\mathbb{Z}_n, B)$ and call it an *alternating sum semigroup* [25]. For example, consider an alternating sum semigroup with letters $B = \{0, 1, 2, 3, 4, 11, 14\}$ in the arithmetic modulo $n = 17$. In this semigroup we have $3 \cdot 1 \cdot 4 \cdot 14 = 1 \cdot 11 \cdot 4 \cdot 2$ because $3 - 1 + 4 - 14 = 1 - 11 + 4 - 2 = 5 \bmod 17$.

To assign useful numerical labels[3] to the arcs of a 4-plat diagram $\mathfrak{d}$, label the two leftmost arcs by 0 and 1, as on the example in Fig. 2. To distinguish between arcs and their labels, we shall denote the label of an arc x by b_x. Propagate the labelling as follows: moving from the left to the right on the diagram, at each crossing $x \dashv y \vdash z$, let $b_z = 2b_y - b_x$. After we have done this at every crossing, two arcs will get two labels each: in our example, the top-right arc on the diagram is labelled -6 and 11, and the bottom-right arc on the diagram is labelled 1 and 18. Considering either of the two equalities $-6 = 11$ or $1 = 18$, we conclude that we should treat the labels as numbers in the arithmetic modulo 17 (hence, for convenience, -3 can be rewritten as 14). Given the modulus n ($n = 17$ in our

[3] The described procedure is a version of so-called Fox coloring [17]. Note that in general, labels of some distinct arcs may coincide.

example) and the set of arc labels B ($B = \{0, 1, 2, 3, 4, 11, 14\}$ in our example), consider an alternating sum semigroup $AS(\mathbb{Z}_n, B)$.

Proposition 2. *$AS(\mathbb{Z}_n, B)$ produced using the procedure above is a factor semigroup of $K\mathfrak{d}$.*

Proof. Consider a mapping from $K\mathfrak{d}$ to $AS(\mathbb{Z}_n, B)$ induced by the rule $x \mapsto b_x$, where x is an arc. The knot semigroup $K\mathfrak{d}$ is defined by relations stating that at each crossing $x \dashv y \vdash z$ we have $xy = yz$ and $yx = zy$. The two corresponding equalities are satisfied in $AS(\mathbb{Z}_n, B)$: indeed, words $b_x b_y$ and $b_y b_z$ both have length 2; the alternating sum of $b_x b_y$ is $b_x - b_y$, and since $b_z = 2b_y - b_x$, the alternating sum of $b_y b_z$ is also $b_x - b_y$; therefore, $b_x b_y = b_y b_z$; similarly, $b_y b_x = b_z b_y$.

The following result[4] is first proved as Proposition 3.2 in [2], or see [14]; the idea originates from [8].

Fact 2. *A knot is a 4-plat knot if and only if its π-orbifold group is dihedral.*

Generalising the 'only if' part of Fact 2 to knot semigroups, we can state the following conjecture (first formulated in [25], after having described knot semigroups of some subclasses of the class of 4-plat knots):

Conjecture 3. The homomorphism from $K\mathfrak{d}$ onto $AS(\mathbb{Z}_n, B)$ described in Proposition 2 is an isomorphism.

3.1 Defining Relations for an Alternative Sum Semigroup

In Subsect. 3.2 we present experiments which use automated reasoning to prove Conjecture 3 for a number of 4-plats. Proving the isomorphism becomes possible if $AS(\mathbb{Z}_n, B)$ is redefined using defining relations. In this subsection we introduce an algorithm for finding a finite list of defining relations for an alternating sum semigroup.

Below we assume that set B contains 0; this is merely a convenience for simpler notation, and all statements can be rewritten to use another element of B instead of 0. All words below are assumed to be words over the alphabet B.

Let us say that a word w is zero-ending if its last letter is 0. For every word w we shall define its canonical form $c(w)$ as the smallest word (relative to the right-to-left dictionary order) which is equal to w in $AS(\mathbb{Z}_n, B)$.

Let us say that a pair of sets of words Y, Z is a *basis* if

1. For every $w \in Y$ its canonical form $c(w)$ is not zero-ending;
2. For every $w \in Z$ its canonical form $c(w)$ is zero-ending;
3. Every word is either contained in Y or has a suffix contained in Z.

[4] We are grateful to José Montesinos (Universidad Complutense de Madrid), Genevieve Walsh (Tufts University) and Vanni Noferini (University of Essex) for attracting our attention to this result.

Theorem 3. *Suppose Y, Z is a basis. Then all equalities of words in $AS(\mathbb{Z}_n, B)$ can be deduced*[5] *from defining relations $w = c(w)$, where $w \in Y \cup Z$.*

Proof. We shall use the proof by induction on the length of words. For words of length 1, there is no need to apply the defining relations because none of words is equal to another word. Now assume that all equalities in $AS(\mathbb{Z}_n, B)$ for words shorter than the length we are considering can be deduced from the defining relations. It is sufficient to prove that for each word v we can deduce the equality $v = c(v)$. Three cases are possible:

1. Suppose $c(v)$ is not zero-ending. Consider v as a product of two words $v = v_1 v_2$ and assume that $c(v_2)$ is zero-ending; then v is equal to a zero-ending word $v_1 c(v_2)$, hence, $c(v)$ is also zero-ending; since it is not so, we conclude that none of suffixes of v has a zero-ending canonical form. Therefore, neither v nor any of its suffixes is contained in in Z. Hence, $v \in Y$, and the equality $v = c(v)$ is one of the defining relations.
2. Suppose $c(v)$ is zero-ending and $v \in Z$. Then the equality $v = c(v)$ is one of the defining relations.
3. Suppose $c(v)$ is zero-ending and $v \notin Z$. Then v is a product of two words $v = v_1 v_2$ such that $v_2 \in Z$. Then $c(v_2)$ is zero-ending, and we can represent it as $c(v_2) = t0$ for some word t. Hence, $v = v_1 t0$. On the other hand, since $c(v)$ is zero-ending, we can represent it as $c(v) = s0$ for some word s. Thus, $v_1 t0 = s0$; since $AS(\mathbb{Z}_n, B)$ is cancellative, $v_1 t = s$. This is an equality of two words whose length is less than that of v; thus, by induction, this equality can be deduced from the defining relations. Therefore, the equality $v = c(v)$ can be deduced in the order $v_1 v_2 = v_1 t0 = s0$, that is, first by applying the defining relation $v_2 = c(v_2)$, and then by applying all the defining relations needed to prove that $v_1 t = s$.

Theorem 3 suggests a simple algorithm for building a basis for a given alternating sum semigroup. Consider all words one after another, starting from the shorter ones; as you consider a word w, add it to Y if $c(w)$ is not zero-ending, or to Z if $c(w)$ is zero-ending, or to neither if a suffix of w is contained in Z. When all possible suffixes of longer words have been added to Z, stop. This algorithm will produce a basis; however, it would be nice to have an assurance that the algorithm will terminate; it is provided by the following statement.

Proposition 3. *Every sufficiently long word in an alternating sum semigroup contains a suffix whose canonical form is zero-ending. Hence, each alternating sum semigroup has a finite basis Y, Z.*

Proof. We shall prove that the canonical form of every word w of length $L \geq 2n^2$ is zero-ending. Indeed, there is a letter, say, a, which stands at least at n

[5] Note that here we mean the usual semigroup deduction, not a more complicated one used in cancellative semigroups. It is useful to remind oneself of this, because knot semigroups are defined using a cancellative presentation, and it makes proving equalities of words in knot semigroups more involved.

distinct even positions in w. Notice that in an alternating sum semigroup we have $xyz = zyx$ for any three letters x, y, z. Applying these equalities as needed, move n letters a to positions $L, L-2, \ldots, L-2n+2$; thus, we produce a word w' which is equal to w such that $w' = vax_1ax_2 \cdots ax_{n-1}a$ for some word v and letters $x_1, x_2, \ldots, x_{n-1}$. Notice that in an alternating sum semigroup $AS(\mathbb{Z}_n, B)$ the word w' is equal to $w'' = v0x_10x_2 \cdots 0x_{n-1}0$. The word w'' is equal to w and is zero-ending; therefore, $c(w)$ is zero-ending.

3.2 Experiments

To test Conjecture 3, we considered 4-plat knot diagrams with up to 9 crossings; we restricted ourselves to canonical 4-plat knot diagrams (see, for example, Proposition 12.13 in [3] and page 187 in [21]), that is, those in which every crossing is either a clockwise half-twist of strands in positions 1 and 2 or an anticlockwise half-twist of strands in positions 2 and 3, and the arcs on the left-hand side of the diagram connect level 1 to level 2 and level 3 to level 4 (like on the diagram in Subsect. 3). Note that a knot semigroup is defined for a diagram and not for a knot, and two diagrams of the same knot can have non-isomorphic semigroups; in our study of Conjecture 3 we consider all individual diagrams. For instance, the number of distinct knots with 9 crossings is 49. However, when we consider canonical 4-plat diagrams with 9 crossings, each of 9 crossings can be in one of two possible positions, between levels 1 and 2 or between levels 2 and 3 on the diagram; in addition to this, the arcs on the right-hand side of the diagram can be connected in two possible ways: level 1 to level 2 and level 3 to level 4, or level 1 to level 4 and level 2 to level 3. Therefore, we generated $2^{9+1} = 1024$ diagrams. Out of these, 664 diagrams were knots, and we confirmed Conjecture 3 for each of them. Other diagrams are links and not knots, and we discard them from consideration; knot semigroups of 4-plat links are not alternating sum semigroups (for example, knot semigroups of a class of 4-plat links is described in Sect. 6 in [25]).

For each diagram $\mathfrak{d}$ out of these 664 diagrams, we produced an alternating sum semigroup $S = AS(\mathbb{Z}_n, B)$ using the procedure described before Proposition 2 (we wrote a Python script to do this). Semigroup S is, according to Proposition 2, a factor semigroup of the knot semigroup $K\mathfrak{d}$. Thus, to prove that $K\mathfrak{d}$ and S are isomorphic, it is sufficient to show that all defining relations of S are derivable in $E_{K\mathfrak{d}}$, the equational theory of $K\mathfrak{d}$.

We wrote another Python script which finds defining relations for a given alternating sum semigroup S using the procedure described in Subsect. 3.1 and outputs the task $E_{K\mathfrak{d}} \vdash E_S$ to be used by Prover9, where E_S means the conjunction of all defining relations of S. The tasks were then passed to Prover9. Due to a large size of E_S, in order to get automated proofs, some tasks had to be split into up to four subtasks $E_{K\mathfrak{d}} \vdash E_S^i$, with $E_S = \cup_i E_S^i$. Eventually all proofs have been obtained with the time limit 1200 s for each task.

4 Future Research

We are continuing working of proving Conjecture 1. Its 'if' part follows from the observation below. The 'only if' part is much more difficult to prove.

Proposition 4. *If $K\mathfrak{d}$ is an infinite cyclic semigroup then $\mathfrak{d}$ represents the trivial knot.*

Proof. Indeed, if $K\mathfrak{d}$ is cyclic then its factor group $O\mathfrak{d}$ is isomorphic to $\mathbb{Z}_2$, whose subgroup $T\mathfrak{d}$ is trivial. Since $T\mathfrak{d}$ is trivial, by Fact 1, $\mathfrak{d}$ represents the trivial knot.

Among algebraic constructions listed in Fact 1, computational experiments with keis [6], quandles [7], semigroups and π-orbifold groups (in this paper) have been conducted. Experiments with groups have been only started in [6], and experiments with two-fold groups (which are smaller and may be easier to manipulate) can be conducted in the future.

Using semigroups or π-orbifold groups to prove that a knot diagram represents the trivial knot is a topic for more future research. As discussed in Subsect. 2.5, this is not the fastest method of proving that a knot is trivial. However, we have reasons to believe that such proofs, if they are produced by a specialised prover and properly presented, can be more human-readable than others (for example, those based on keis). We shall continue studying such proofs because of new mathematical constructions arising in them and because this is an impressive example of how complicated computer-generated proofs can be made human-readable.

5 Technical Details

We used Prover9 and Mace4 version 0.5 (December 2007) [19] and one of two system configurations:

(1) AMD A6-3410MX APU 1.60 Ghz, RAM 4 GB, Windows 7 Enterprise when producing Tables 1 and 3 (and results from [6] used in Subsects. 2.3 and 2.5);
(2) Intel(R) Core(TM) i7-4790 CPU 3.60 Ghz, RAM 32 GB, Windows 7 Enterprise when producing Table 2 and results in Sect. 3.

We have used default iterative Mace4 search strategy, except for the cases explicitly mentioned as using different strategies in Subsects. 2.3 and 2.4. We have used default search strategies in Prover9, with the following exceptions. For the results presented in Subsect. 2.2 we have used Knuth-Bendix term ordering (KBO) instead of default choice of LPO (Lexicographic Path Ordering). In order to handle large clauses occurring in the proofs reported in Subsect. 3.2 we have set `max_weight` (maximum weight of clauses) to 8000.

We have published all computer-generated proofs online[6].

[6] https://zenodo.org/record/1009577, https://doi.org/10.5281/zenodo.1009577.

References

1. Birkhoff, G.: On the structure of abstract algebras. In: Mathematical Proceedings of the Cambridge Philosophical Society, vol. 31, pp. 433–454. Cambridge University Press (1935)
2. Boileau, M., Zimmermann, B.: The π-orbifold group of a link. Mathematische Zeitschrift **200**(2), 187–208 (1989)
3. Burde, G., Heusener, M., Zieschang, H.: Knots. De Gruyter, Berlin (2013)
4. Caferra, R., Leitsch, A., Peltier, N.: Automated Model Building, vol. 31. Springer, Dordrecht (2013). https://doi.org/10.1007/978-1-4020-2653-9
5. Elhamdadi, M., Nelson, S.: Quandles, vol. 74. American Mathematical Society, Providence (2015)
6. Fish, A., Lisitsa, A.: Detecting unknots via equational reasoning, I: exploration. In: Watt, S.M., Davenport, J.H., Sexton, A.P., Sojka, P., Urban, J. (eds.) CICM 2014. LNCS (LNAI), vol. 8543, pp. 76–91. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08434-3_7
7. Fish, A., Lisitsa, A., Stanovský, D.: A combinatorial approach to knot recognition. In: Horne, R. (ed.) EGC 2015. CCIS, vol. 514, pp. 64–78. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25043-4_7
8. Fox, R.: A quick trip through knot theory. In: Fort, M.K. (ed.) Topology of Three-Manifolds. Prentice-Hall, Englewood Cliffs (1962)
9. Gilbert, N.D., Porter, T.: Knots and Surfaces. Oxford University Press, New York (1994)
10. Huet, G., Oppen, D.C.: Equations and rewrite rules. In: Book, R.N. (ed.) Formal Language Theory: Perspectives and Open Problems, pp. 349–405. Academic Press, New York (1980)
11. Joyce, D.: A classifying invariant of knots, the knot quandle. J. Pure Appl. Algebra **23**(1), 37–65 (1982)
12. Kauffman, L.H., Henrich, A.: Unknotting unknots. https://arxiv.org/abs/1006.4176
13. Kauffman, L.H., Lambropoulou, S.: Hard unknots and collapsing tangles. https://arxiv.org/abs/math/0601525v5
14. Kawauchi, A.: A Survey of Knot Theory. Birkhäuser, Basel (1996)
15. Lackenby, M.: Upper bound on Reidemeister moves. Ann. Math. **182**, 1–74 (2015)
16. Raymond-Lickorish, W.B.: An Introduction to Knot Theory, vol. 175. Springer, New York (1997). https://doi.org/10.1007/978-1-4612-0691-0
17. Livingston, C.: Knot Theory, vol. 24. Cambridge University Press, Cambridge (1993)
18. Manturov, V.: Knot Theory. CRC Press, Boca Raton (2004)
19. McCune, W.: Prover9 and Mace4 (2005–2010). http://www.cs.unm.edu/mccune/prover9/
20. Morgan, J.W., Bass, H. (eds.): The Smith Conjecture. Elsevier, Amsterdam (1984)
21. Murasugi, K.: Knot Theory and Its Applications. Springer, Boston (1996). https://doi.org/10.1007/978-0-8176-4719-3_15
22. Ochiai, M.: Non-trivial projections of the trivial knot. http://repository.kulib.kyoto-u.ac.jp/dspace/handle/2433/99940
23. O'Hara, J.: Energy of knots and infinitesimal cross ratio. Geom. Topol. Monogr. **13**, 421–445 (2008)

24. Unknot. https://en.wikipedia.org/wiki/Unknot
25. Vernitski, A.: Describing semigroups with defining relations of the form xy = yz and yx = zy and connections with knot theory. Semigroup Forum **95**(1), 66–82 (2017)
26. Winker, S.K.: Quandles, knot invariants, and the n-fold branched cover. Ph.D. thesis, University of Illinois at Chicago (1984)

Balancing Expression Dags for More Efficient Lazy Adaptive Evaluation

Martin Wilhelm(✉)

Institut für Simulation und Graphik, Otto-von-Guericke-Universität Magdeburg,
Universitätsplatz 2, 39106 Magdeburg, Germany
martin.wilhelm@ovgu.de

Abstract. Arithmetic expression dags are widely applied in robust geometric computing. In this paper we restructure expression dags by balancing consecutive additions or multiplications. We predict an asymptotic improvement in running time and experimentally confirm the theoretical results. Finally, we discuss some pitfalls of the approach resulting from changes in evaluation order.

1 Introduction

Most theoretical algorithms in the field of computational geometry are based on the real RAM model of computation and therefore assume exact real number arithmetic at unit cost. Actual processors cannot represent real numbers and instead use floating-point arithmetic. Computing exact numerical values is expensive and not always possible. Luckily, it is also seldom necessary to compute exact values in order to ensure robustness in geometric algorithms. The Exact Geometric Computation paradigm instead only demands that the decisions made in a geometrical algorithm are correct [9,10]. A common technique used for exact-decisions computation is to store the computation history in an arithmetic expression dag and then adaptively (re)compute the result with a higher precision until a verified decision can be made. Several expression-dag-based number types have been developed with different evaluation strategies. Strategies can be to gradually increase the precision bottom-up (`LEA` [1]) or fall back to exact computation (`CGAL::Lazy_exact_nt` [8]) if a decision cannot be verified, or to use a precision-driven[1] evaluation (`leda::real` [2], `Core::Expr` [5,11], `Real_algebraic` [6]). All of the mentioned number types suffer from high performance overhead compared to standard floating-point arithmetic.

In this work we make an attempt to improve the performance of dag-based number types by restructuring the underlying expression dag in certain situations. Restructuring the expression dag was originally proposed by Yap [10]. To our knowledge, there is no previous work that actually implements any restructuring strategy. We focus on reducing the depth of an expression dag, i.e. the size

[1] This should more correctly be called "accuracy-driven", but we use the term "precision-driven" throughout this paper for historical reasons.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 19–33, 2017.
https://doi.org/10.1007/978-3-319-72453-9_2

of the longest path from any node to the root. The accuracy needed at a node in an expression dag to guarantee a certain error at its root generally increases with the size of the longest path between the node and the root. Therefore a decrease in expression depth can be expected to lead to better error bounds at lower precision and, consequently, to a better performance of dag-based number types. We restructure the expression dag by "balancing" consecutive additions and consecutive multiplications, such that the maximum depth of the involved operators is minimized. By this we also increase the independence of the nodes, which makes it more feasible to parallelize the evaluation. In this work, however, we will not elaborate on advantages regarding parallelization.

We provide a theoretical analysis and evaluate our strategy based on the number type `Real_algebraic` introduced by Mörig et al. [6].

2 Theoretical Foundation

An *expression dag* is a rooted ordered directed acyclic graph, which is either

1. A single node containing a number or
2. A node representing a unary operation $\{\sqrt{}, -\}$ with one, or a binary operation $\{+, -, *, /\}$ with two, not necessarily disjoint, expression dags as children.

We call an expression dag E' whose root is part of another expression dag E a *subexpression of* E.

Let E be an expression dag, let $\circ \in \{+, *\}$ and let E' be a subexpression of E with root r of type $\circ$. Let T be a connected subgraph of E', containing r, such that all nodes in $T - r$ have at most one predecessor in E and are of type $\circ$. Then T is a tree and we call T an *operator tree*. The children of the leaves of T in E are called *operands* of T. We restructure E by replacing all maximal operator trees in E by a balanced operator tree with the same number of operands. For a single tree, we call this replacement *balancing the operator tree*. If all maximal operator trees in E are replaced, we call the process *balancing the expression dag*.

We determine the asymptotic running time of a single precision-driven evaluation before and after balancing the expression dag for a series of additions or a series of multiplications. Assumptions on the unit costs for the arithmetic operations and the increase in accuracy are consistent with `leda::real`, `Real_algebraic` and partly with `Core::Expr`.

2.1 Addition

Assume we have a dag-based number type that determines the result of an addition $z = x + y$ with absolute accuracy q in time $\Theta(q + \log|z|)$ if x and y are accurate up to $q + c$ fractional digits, where c is some constant.

Let $x_1, ..., x_n$ be distinct floating point numbers with exponent $\leq e, e \geq 0$. We want to determine the running time to compute $z = \sum_{i=1}^{n} x_i$ with absolute accuracy q. Any expression dag for z contains an operator tree consisting of all addition nodes. Assume that the operator tree is a linear list, i.e. the computation

order is equivalent to $x_1 + (x_2 + (x_3 + ... + (x_{n-1} + x_n)))$. Then the i-th addition (counting from the root) must be accurate up to $q_i = q + ic$ fractional digits and the magnitude of its result is at most $e_i = e + \lceil \log(n-i) \rceil$. Therefore we get the time for computing z by adding the time needed on each level as

$$\begin{aligned} T_{\text{list}}(q) &= O\left(\sum_{i=0}^{n-1}(q_i + e_i)\right) = O\left(\sum_{i=0}^{n-1}(q + ic + e + \log(n-i))\right) \\ &= O\left(nq + n^2 + ne\right) \end{aligned}$$

This bound is tight if all summands have maximum exponent. Now assume the operator tree is perfectly balanced, i.e. the computation order is equivalent to

$$((((x_1 + x_2) + (x_3 + x_4)) + ...) + (... + ((x_{n-3} + x_{n-2}) + (x_{n-1} + x_n))))$$

Then at level i there are 2^i additions, which must be accurate up to $q_i = q + ic$ fractional digits. The magnitude of their result is at most $e_i = e + \log n - i$. So the asymptotic bound for the computation time shrinks to

$$T_{\text{bal}}(q) = O\left(\sum_{i=0}^{\log n} 2^i (q + ic + e + \log n - i)\right) = O\left(nq + n \log n + ne\right)$$

2.2 Multiplication

For multiplication we assume the number type computes the result of $z = x * y$ with absolute accuracy q in time $\Theta((q + \log|z|)^{\log 3})$ if x is accurate up to $q + c + \lceil \log|y| \rceil$ and y up to $q + c + \lceil \log|x| \rceil$ fractional digits, where c is some constant. We determine the running time to compute $z = \prod_{i=1}^{n} x_i$ with absolute accuracy q.

We consider the operator tree consisting of all multiplication nodes in an expression dag for z. Let $e \geq 0$ be the maximum exponent of $x_1, ..., x_n$. In the unbalanced case the accuracy needed increases by at most $c + e$ with each level top-down, whereas the maximum exponent of the result increases by e bottom-up. Assuming that $x_1, ..., x_n$ are exact, we do not need to increase the accuracy of the leaves. Then we get

$$\begin{aligned} T_{\text{list}}(q) &= O\left(\sum_{i=0}^{n-1}(q_i + e_i)^{\log 3}\right) = O\left(\sum_{i=0}^{n-1}(q + i(c+e) + (n-i)e)^{\log 3}\right) \\ &= O\left(nq^{\log 3} + n^{\log 3 + 1} + n^{\log 3 + 1} e^{\log 3}\right) \end{aligned}$$

This bound is tight if $x_1, ..., x_n$ all have exponent e. When the operator tree is balanced, the accuracy needed increases by $c + e_{i+1}$ at level i, where $e_i = 2^{\log n - i} e$, so the requested accuracy at level i is

$$q_i = q + ic + \sum_{j=0}^{i+1} 2^{\log n - j} e \leq q + ic + 2^{\log n + 1} e$$

Therefore

$$T_{\text{bal}}(q) = O\left(\sum_{i=0}^{\log n} 2^i (q + ic + 2^{\log n+1} e + 2^{\log n - i} e)^{\log 3}\right)$$
$$= O\left(nq^{\log 3} + n(\log n)^{\log 3} + n^{\log 3+1} e^{\log 3}\right)$$

If $e > 0$ the improvement we get from balancing the tree is dominated by the cost for managing the increasing number of integer digits. If one can expect the exponent to be bounded from above, the improvement gets asymptotically significant. Let $e_{\max}$ be the largest exponent occurring during the whole computation. Then

$$T_{\text{list}}(q) = O\left(\sum_{i=0}^{n-1} (q + i(c + e_{\max}) + e_{\max})^{\log 3}\right)$$
$$= O\left(nq^{\log 3} + n^{\log 3+1} + n^{\log 3+1} e_{\max}^{\log 3}\right)$$

whereas

$$T_{\text{bal}}(q) = O\left(\sum_{i=0}^{\log n} 2^i (q + i(c + e_{\max}) + e_{\max})^{\log 3}\right)$$
$$= O\left(nq^{\log 3} + n(\log n)^{\log 3} + n(\log n)^{\log 3} e_{\max}^{\log 3}\right)$$

The asymptotic bound for $T_{\text{list}}(q)$ is tight if the values of $\Theta(n)$ inner nodes are of order $e_{\max}$.

3 Implementation

The balancing strategy has been implemented for the dag-based exact-decision numbertype `Real_algebraic` designed by Mörig et al. [6]. This number type consists of a single- or multi-layer floating-point-filter [4], which falls back to adaptive evaluation with bigfloats stored in an expression dag [3]. Balancing is done at most once at each node, right before the first bigfloat evaluation. Otherwise, existing results would have to be recomputed after changing the structure of the dag, which could potentially lead to a massive overhead if subexpressions need to be evaluated during dag construction. Once evaluated nodes are therefore treated as operands in any subsequent balancing process.

We call evaluations of subexpressions of an expression dag E *partial evaluations of* E. By preventing evaluated nodes to be part of another operator tree, frequent partial evaluations during construction can fully negate the benefits of the balancing strategy. If partial evaluations occur only sporadically, then their impact on the *expression depth of* E, i.e. the maximum distance of any node to its root, is small, since each of the involved subexpressions have been balanced when they were evaluated for the first time.

Observation 1. *Let E be an expression dag consisting of n additions or n multiplications and $n+1$ bigfloats. Let d be the expression depth of E after its evaluation. If at most k partial evaluations of E occur before the evaluation of E, then $d \leq k\lceil \log \frac{n}{k} \rceil$.*

On the first evaluation of a node that cannot be handled by the floating-point-filter, the balancing process starts at this node recursively. If the current node contains an addition or a multiplication and has not be balanced before, all operands of the maximal operator tree containing this node as root will be retrieved. If the depth of the operator tree can be reduced, it gets balanced (cf. Algorithm 1). For almost-balanced trees a slight decrease in depth may not justify restructuring a large tree. Therefore it might be useful to experimentally decide on a factor tightening this condition in later implementations.

Algorithm 1. The relevant operator trees are retrieved in the form of an operand list with an associated depth. By comparing the depth with the number of operands it gets decided whether the trees should be balanced.

```
Data: current node node
if node is not balanced then
    if node is addition or multiplication then
        (operands, depth) = retrieve_operands(node)
        if depth > ⌈log |operands|⌉ then
            balance current operation
        end
        foreach op ∈ operands do
            recurse on op
        end
    else
        recurse on children
    end
    mark node as balanced
end
```

The operands get retrieved through a depth-first search. Nodes can be retrieved more than once and therefore the same node can represent multiple operands. A node is treated as an operand if one of the following conditions holds:

1. The node is not of the same type as the current operation (i.e. $+$ or $*$).
2. The node has already been balanced (and therefore initialized).
3. The node has more than one parent.

The third condition is necessary, since the subexpression represented by this node will be destroyed during balancing. If a full copy of the node would be created instead, this may lead to an exponential blowup for highly self-referential structures (cf. Fig. 1).

A similar observation as for the second condition (cf. Observation 1) can be made. If few operator nodes have more than one parent, the overall impact on the expression depth is small.

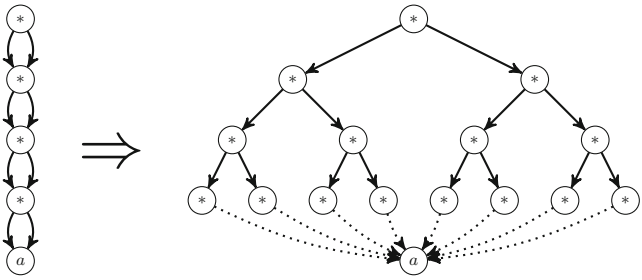


Fig. 1. An expression dag computing a^{16} and its exponential expansion that results from resolving multiple references through copying.

Observation 2. *Let E be an expression dag consisting of n additions or n multiplications and $n+1$ bigfloats. Let d be the expression depth of E after its evaluation. If at most k operator nodes of E have more than one reference, then $d \leq k\lceil \log \frac{n}{k} \rceil$.*

We balance an operation by combining two operands to a new operand until only one node is left by treating the operand vector like a queue (cf. Algorithm 2). Note that this strategy does not preserve the evaluation order of the operands if the number of operands is not a power of two. This can have consequences for the running time and may obfuscate the experiments. If operand order is of importance, it can be preserved by inserting dummy nodes with values 0 for addition and 1 for multiplication up to the next power of two.

Algorithm 2. The operator tree is restructured by discarding all nodes except the root and building a new balanced operator tree bottom-up by repeatedly combining the two smallest subtrees to a new tree.

Data: operand vector *operands*, operation type $\circ$, root node *root*
$size = operands.size()$;
for $i = 0$ ***to*** $size - 2$ **do**
  | $operands.add(new\ Node(operands[2i], operands[2i+1], \circ))$
end
$root.left = operands[2 * size - 2]$;
$root.right = operands[2 * size - 1]$;

4 Experiments

All experiments are run on an Intel Core i5 660 with 8 GB RAM under Ubuntu 16.04 LTS. We use Boost interval arithmetic as floating-point-filter and MPFR bigfloats for the bigfloat arithmetic. The code is compiled using g++ 5.4.0 with C++11 on optimization level `O3` and linked against Boost 1.62.0 and MPFR 3.1.0. Test results are averaged over 25 runs each if not stated otherwise. The variance for each data point is negligible.

We will perform two simple experiments to evaluate our strategy. In our first experiment we compute the sum of the square roots of the natural numbers 1 to n with accuracy q.

```
template <class NT> void sum_of_sqrts(const int n, const long q){
  NT sum = NT(0);
  for (int i = 1; i <= n; ++i) {
    sum += sqrt(NT(i));
  }
  sum.guarantee_absolute_error_two_to(q);
}
```

The second test computes the generalized binomial coefficient

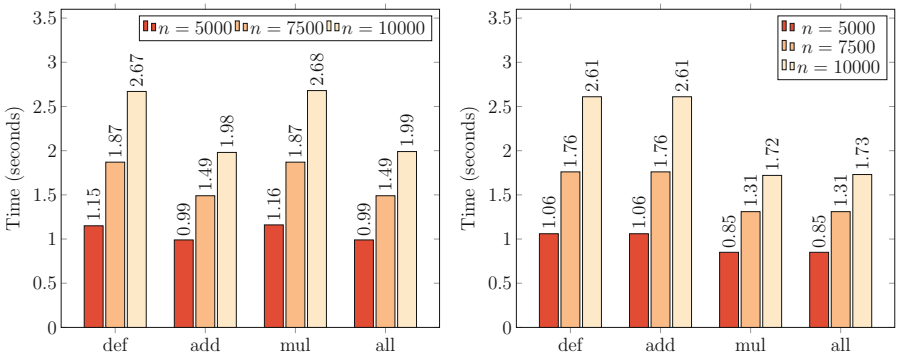
$$\binom{\sqrt{13}}{n} = \frac{\sqrt{13}(\sqrt{13}-1)\cdots(\sqrt{13}-n+1)}{n(n-1)\cdots 1}$$

with accuracy q.

```
template <class NT> void bin_coeff(const int n, const long q){
  NT b = sqrt(NT(13));
  NT num = NT(1); NT denom = NT(1);
  for (int i = 0; i < n; ++i) {
    num *= b - NT(i);
    denom *= NT(i+1);
  }
  NT bc = num/denom;
  bc.guarantee_absolute_error_two_to(q);
}
```

For each test we compare four different implementations. We distinguish between no balancing (def), balancing only addition (add), balancing only multiplication (mul) and balancing both addition and multiplication (all).

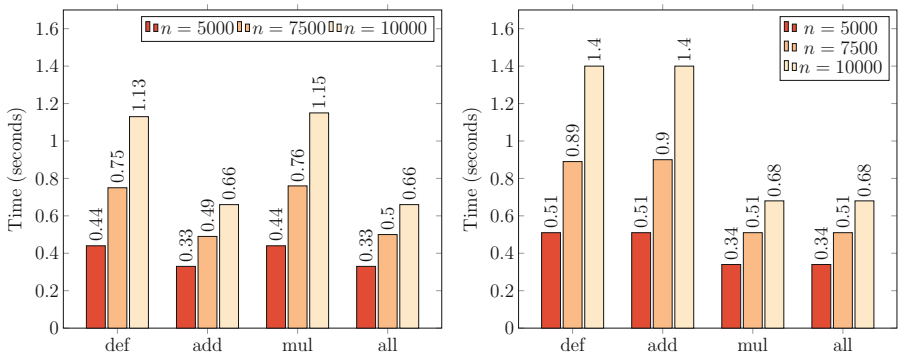
The sum-of-square-roots test as well as the binomial coefficient test provide simple examples for when balancing can be of use (cf. Fig. 2). Obviously balancing multiplication does not have a positive effect on the sum-of-square-roots test



(a) Sum-of-square-roots (b) Binomial coefficient

Fig. 2. Performance gain through balancing for `sum_of_sqrts` and `bin_coeff` with a requested accuracy of $q = 50000$ for different values of n.

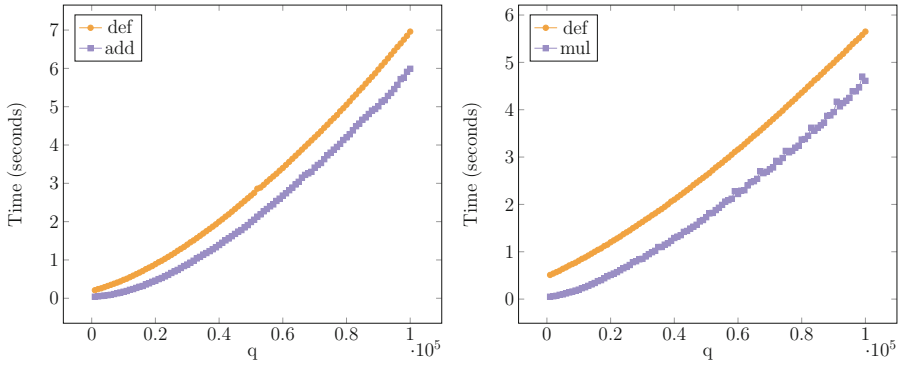
and balancing addition does not have a positive effect on the binomial coefficient computation. There is a small overhead in these cases due to the traversal of the dag. The overhead vanishes in `all`, since the same procedure is used for both addition and multiplication.



(a) Sum-of-square-roots (b) Binomial coefficient

Fig. 3. Performance gain through balancing for `sum_of_sqrts` and `bin_coeff` with a requested accuracy of $q = 25000$. The relative gain is larger than for $q = 50000$ (cf. Fig. 2).

The relative benefit of balancing increases if the precision increase due to the number of operands is large relative to the requested accuracy for the result. Figure 3 shows the performance gain through balancing for a requested accuracy of $q = 25000$. With 10000 operands, the relative gain is about 42% for `sum_of_sqrts` and 51% for `bin_coeff` compared to 26% and 34% for $q = 50000$. The theoretical analysis from Sect. 2 predicts that the absolute performance gain



(a) Sum-of-square-roots (b) Binomial coefficient

Fig. 4. Absolute performance gain through balancing for `sum_of_sqrts` and `bin_coeff` for $n = 10000$ with different requested accuracies (average over five runs). The absolute gain is almost independent of q. The relative gain decreases.

primarily depends on the number of the operands that can be balanced and is independent from the requested accuracy. The experimental results largely confirm this assumption as shown in Fig. 4. Since balancing is done before the first evaluation, the overhead due to the balancing procedure only depends on the size of the expression dag and the number of operands.

5 Caveats

When restructuring an expression dag there are some potential pitfalls one should be aware of. Changing the structure of an expression dag leads to a change in evaluation order, which may in turn influence the performance. Other hurdles are even more subtle, since they result from implementation details of the underlying bigfloat arithmetic. We show examples, where this leads to problems for balancing. However, the caveats are not restricted to balancing, but apply to restructuring attempts in general.

5.1 Evaluation Order

When evaluating a dag-based number type recursively, a slight change in expression order can have an unexpectedly high impact on the evaluation time [7]. Balancing the dag may have a negative impact on the optimal expression order. One example where this may occur is the computation of the geometric sum $\sum_{i=0}^{n} r^i$ with $r < 1$.

```
template <class NT> void geometric_sum(const int n, const long q){
  NT r = sqrt(NT(13)/NT(64));
  NT ri = NT(1); NT s = ri;
  for (int i=0; i<n; ++i){
    ri *= r;
    s  += ri;
  }
  s.guarantee_absolute_error_two_to(q);
}
```

We call the multiplication node m_i resulting from the i-th multiplication *deeper* than the node m_j resulting from the j-th multiplication if $i < j$ and *shallower* if $j < i$. If m_j is shallower than m_i then m_j is an ancestor of m_i in the expression dag. When balancing the expression dag the accuracy needed at the deeper multiplication nodes decreases, while the accuracy needed at the shallower nodes increases. Since in `geometric_sum` the shallower multiplication nodes depend on the deeper ones, the balancing actually increases the final accuracy needed at the deeper multiplication nodes by an amount logarithmic in the total number of additions. To make things worse, the deeper nodes are still evaluated first (with low precision) and therefore need to be recursively re-evaluated for every shallower multiplication node, leading to a quadratic number of evaluations (Fig. 5).

Note, that this does not happen for the linear computation order if we assume the following increase in accuracy (cf. Sect. 2):

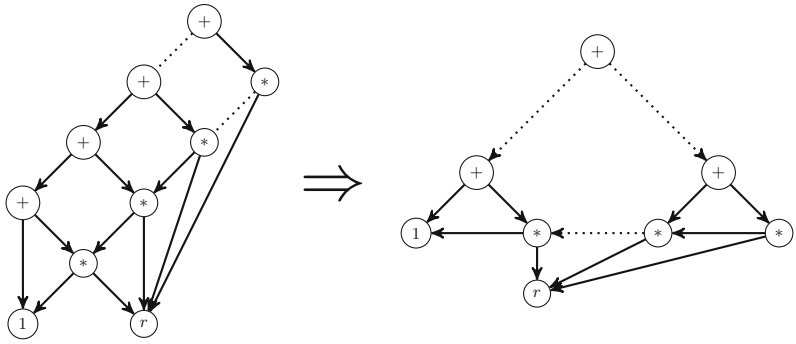


Fig. 5. Expression dags for `geometric_sum` before and after balancing. After balancing, all multiplication nodes are on the same level, with the deeper ones evaluated first, inducing a quadratic number of evaluation steps.

- To evaluate $z = x + y$ with accuracy q, both x and y must be accurate up to $q + 2$ digits.
- To evaluate $z = x * y$ with accuracy q, x must be accurate up to $q + 2 + \lceil \log |y| \rceil$ and y must be accurate up to $q + 2 + \lceil \log |x| \rceil$ digits.

Since for $r < 1$ also $r^i \leq 1$ the increase in accuracy is the same for addition and multiplication. Therefore with linear computation order the multiplication nodes do not need to be re-evaluated after their initial evaluation. If $r > 1$ the linear dag and the balanced dag show similar behavior.[2]

To avoid extensive recomputations, we can compute a topological order and determine the final accuracy needed at each node before recomputing it [7]. We implement this strategy and compare it with recursive evaluation. The standard recursive evaluation procedure essentially works as depicted in Algorithm 3. At each node the needed accuracy of its children is ensured and the value at this node gets recomputed. Nodes can get recomputed several times if they have more than one parent.

Algorithm 3. Evaluating an expression dag by recursively increasing the accuracy of the children before recomputing the current operation.

```
Data: requested accuracy q
if error is larger than 2^-q then
    compute needed accuracy for children
    recurse on children with their respective accuracy
    recompute
end
```

When evaluating topologically we determine a topological order for all inexact nodes and compute the maximum accuracy needed for those nodes.

[2] `Real_algebraic` usually overestimates the exponent by one, therefore in our tests r is chosen to be smaller than 0.5.

Afterwards we recompute the nodes with their maximum accuracy (cf. Algorithm 4). By following this procedure we can guarantee that no node is recomputed more than once during one evaluation of the expression dag.

Algorithm 4. Evaluating an expression dag by finding a topological order and determining the maximum accuracy needed at each node before recomputing them.

```
Data: requested accuracy q
if error is larger than 2^-q then
    top = all inexact nodes in topological order
    for i = 1 to |top| do
        update the required error for the children of top[i]
    end
    for i = |top| downto 1 do
        if top[i].error > top[i].requested_error then
            recompute top[i]
        end
    end
end
```

We execute the geometric sum experiment with the four balancing strategies from before. Furthermore for each of these strategies we evaluate either recursively (r) or in topological order (t).

As the results in Fig. 6 show, balancing the expression dag destroys a favorable evaluation order when computing the geometric sum. Switching to a topological evaluation order negates this effect. Note that the performance loss due

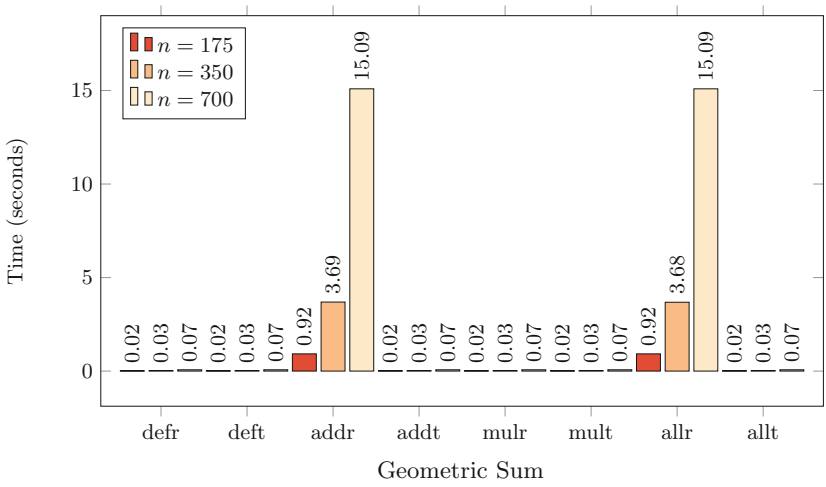
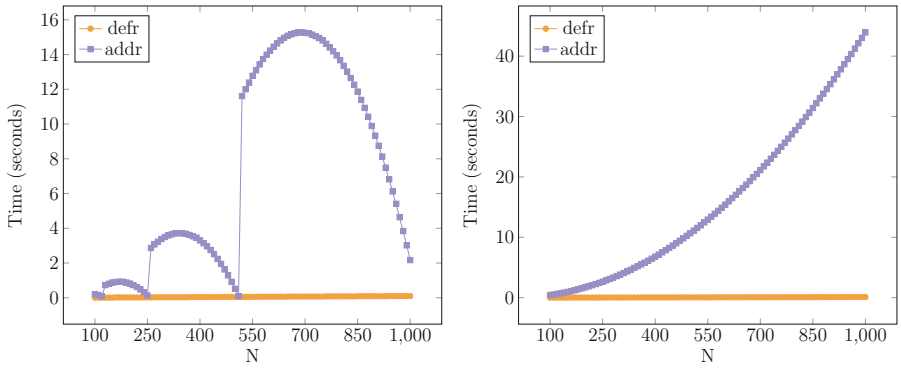


Fig. 6. Balancing additions leads to a massive increase in running time for `geometric_sum` with $q = 50000$ by creating a bad evaluation order. Topological evaluation solves the problem.



(a) Geometric Sum (b) Geometric Sum + OPB

Fig. 7. Comparison of the behavior of `geometric_sum` for increasing n with the balancing procedure from Algorithm 2 and with order preserving balancing (OPB). The original procedure leads to jumps in running time, while order preserving balancing produces the expected quadratic behavior ($q = 50000$, averaged over 5 runs each).

to the logarithmic increase of precision in the balanced case is too small to show in the measurements.

The values for n have been chosen to show spikes in the running time. As pointed out in Sect. 3 our balancing algorithm does not necessarily preserve the order of the operands. If the shallowest multiplication node is evaluated first, this leads to an optimal evaluation order. Figure 7 compares our implementation with an order-preserving balancing strategy.[3]

The algorithm we use to build a balanced tree results in large jumps when stepping from $2^k - 1$ to 2^k operands ($k \in \mathbb{N}$). With $2^k - 1$ operands the previously rightmost operand, i.e. the shallowest multiplication node, becomes the leftmost operand in the balanced tree and therefore the evaluation order is optimal. With 2^k operands the previous operand order is preserved by the algorithm and is therefore the worst possible. If preserving order is enforced, the quadratic increase in running time is evident.

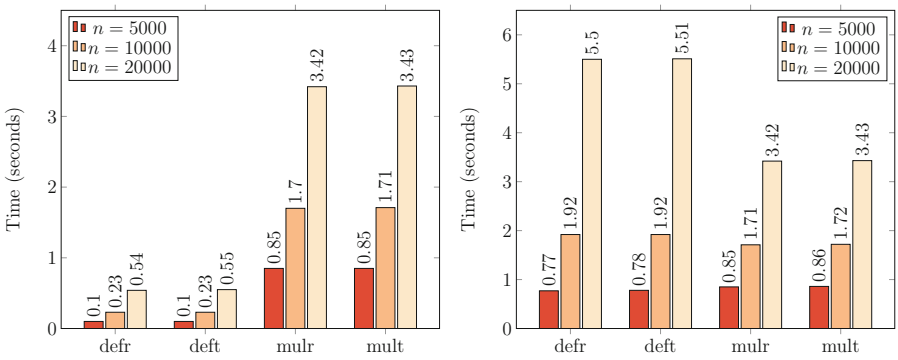
5.2 Operands Matter

In some cases balancing can destroy a favorable dag structure independently from the evaluation order. We compute the telescoping product $\prod_{i=1}^{n-1} \frac{i+1}{i}$ through the following algorithm.

```
template <class NT>
void telescoping_product(const int n, const long q){
  NT prod = NT(1);
  for (int i = 1; i < n; ++i) {
    prod *= NT(i+1)/NT(i);
  }
  prod.guarantee_absolute_error_two_to(q);
}
```

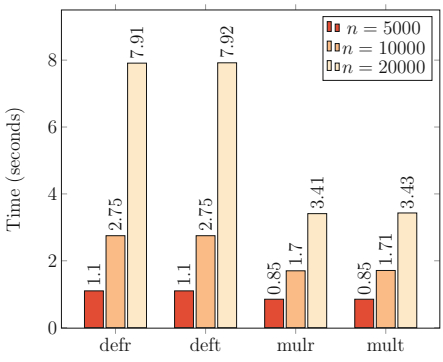
[3] This is implemented by inserting dummy nodes up to the next power of two.

In the experimental results shown in Fig. 8a, a performance decrease due to balancing is evident, which also cannot be corrected through a change in evaluation order. The reason for this effect is that the naive order enables the bigfloat arithmetic to make use of eliminating factors. Bigfloat multiplications involving integers[4] are less expensive[5]. In the original expression order the result of each multiplication is an integer and can be determined as such. Therefore, although significantly reducing the average need of accuracy, balancing has a negative effect on the performance.



(a) Telescoping Product $i \geq 1$

(b) Telescoping Product $i \geq 3$



(c) Telescoping Product Reverse

Fig. 8. Performance of different variants of `telescoping_product` before and after balancing multiplications ($q = 50000$). In the original version balancing destroys a favorable order of the operands, which cannot be corrected for by switching to topological evaluation. When starting with $i = 3$, the order before balancing is less favorable and the performance gain of balancing outweighs the loss for larger n. For `telescoping_product_reverse` no favorable order is destroyed and balancing shows the expected net benefit.

[4] Or integers divided by a power of two.

[5] This behavior was confirmed with both `mpfr` and `leda` bigfloats.

If the product is computed starting with $i = 3$ only every third subexpression evaluates to an integer. While there are still some favorable structures getting disrupted by balancing the expression dag, the benefit of balancing surpasses the loss as the number of operands increases (cf. Fig. 8b). The effect vanishes if the product is computed in reverse order as depicted in the following algorithm.

```
template <class NT>
void telescoping_product_reverse(const int n, const long q){
  NT prod = NT(1);
  for (int i = n-1; i >= 1; --i) {
    prod *= NT(i+1)/NT(i);
  }
  prod.guarantee_absolute_error_two_to(q);
}
```

By this, none of the subexpressions involved evaluates to an integer and only a logarithmic amount of subexpressions evaluates to an integer divided by a power of two. The results of the experiment for the reverse case are shown in Fig. 8c. Now balancing has the expected positive effect on the overall performance. Note that, as expected, the forward loop starting with $i = 3$ without balancing takes approximately two third of the time of the reverse case.

5.3 Overhead

In all tests, except the telescoping product test with $i \geq 1$, the overhead for the balancing procedure as well as for the topological sorting was (usually much) less than 0.5% of the final running time. The running time of `telescoping_product` is unusually small compared to its number of operations, therefore the relative overhead of additional computations is higher. In this case the overhead amounts to less than 2% for balancing and less than 3% for topological sorting.

6 Conclusion

Balancing additions and multiplications in an expression dag can significantly reduce the computation time needed as demonstrated by the sum-of-square-roots test and the binomial coefficient test. The experimental data indicates that the overhead due to the balancing algorithm is small compared to the cost of the bigfloat operations. Balancing may cause changes in the evaluation order that lead to increased running time. Those issues can partially be addressed by switching to a topological evaluation, which can be done with small overhead as well.

We conclude that it is useful to provide a number type supporting balancing of expression dags in combination with topological evaluation. The use of this number type should be considered whenever an algorithm performs a large number of consecutive additions or multiplications. Switching a number type is usually less time-consuming than a deep analysis and adjustment of the used algorithm.

7 Future Work

In this paper we restricted restructuring of the dag to balancing additions and multiplications. Performance increase due to further restructuring is imaginable. Subtractions could easily be included in the balancing process by treating them like an addition and a negation and propagating the negations to the operands. It may also be useful to incorporate divisions into the multiplication balancing process. Since inversions are much more expensive than negations, it seems not feasible to replace them by a multiplication and an inversion. Instead a promising strategy might be to reduce the number of divisions by raising them to the root.

Balancing an expression dag makes its nodes more independent and therefore makes it more accessible for parallelization. Further restructuring with the goal of faster parallelization, e.g. expanding products, might be profitable.

References

1. Benouamer, M.O., Jaillon, P., Michelucci, D., Moreau, J.: A lazy exact arithmetic. In: Proceedings of the 11th Symposium on Computer Arithmetic, pp. 242–249 (1993)
2. Burnikel, C., Mehlhorn, K., Schirra, S.: The leda class real number (1996)
3. Dubé, T., Yap, C.: A basis for implementing exact geometric algorithms (extended abstract) (1993)
4. Fortune, S., van Wyk, C.J.: Efficient exact arithmetic for computational geometry. In: Proceedings of the Ninth Annual Symposium on Computational Geometry, pp. 163–172 (1993)
5. Karamcheti, V., Li, C., Pechtchanski, I., Yap, C.: A core library for robust numeric and geometric computation. In: Proceedings of the Fifteenth Annual Symposium on Computational Geometry, SoCG, pp. 351–359 (1999)
6. Mörig, M., Rössling, I., Schirra, S.: On design and implementation of a generic number type for real algebraic number computations based on expression dags. Math. Comput. Sci. **4**(4), 539–556 (2010)
7. Mörig, M., Schirra, S.: Precision-driven computation in the evaluation of expression-dags with common subexpressions: problems and solutions. In: Kotsireas, I.S., Rump, S.M., Yap, C.K. (eds.) MACIS 2015. LNCS, vol. 9582, pp. 451–465. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32859-1_39
8. Pion, S., Fabri, A.: A generic lazy evaluation scheme for exact geometric computations. Sci. Comput. Program. **76**(4), 307–323 (2011)
9. Schirra, S.: Robustness and precision issues in geometric computation. In: Sack, J.R., Urrutia, J. (eds.) Handbook of Computational Geometry, pp. 597–632. Elsevier, Amsterdam (2000)
10. Yap, C.: Towards exact geometric computation. Comput. Geom. **7**, 3–23 (1997)
11. Yu, J., Yap, C., Du, Z., Pion, S., Brönnimann, H.: The design of core 2: a library for exact numeric computation in geometry and algebra. In: Fukuda, K., Hoeven, J., Joswig, M., Takayama, N. (eds.) ICMS 2010. LNCS, vol. 6327, pp. 121–141. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15582-6_24

Certification Using Newton-Invariant Subspaces

Jonathan D. Hauenstein(✉)

Department of Applied and Computational Mathematics and Statistics,
University of Notre Dame, Notre Dame, IN 46556, USA
hauenstein@nd.edu
http://www.nd.edu/~jhauenst

Abstract. For a square system of analytic equations, a Newton-invariant subspace is a set which contains the resulting point of a Newton iteration applied to each point in the subspace. For example, if the equations have real coefficients, then the set of real points form a Newton-invariant subspace. Starting with any point for which Newton's method quadratically converges to a solution, this article uses Smale's α-theory to certifiably determine if the corresponding solution lies in a given Newton-invariant subspace or its complement. This approach generalizes the method developed in collaboration with F. Sottile for deciding the reality of the solution in the special case that the Newton iteration defines a real map. A description of the implementation in `alphaCertified` is presented along with examples.

Keywords: Newton's method · Certified solutions · Alpha theory
Real solutions · Numerical algebraic geometry

2010 Mathematics Subject Classification: Primary 65G20
Secondary 65H10 · 14Q99

1 Introduction

The increased computing capability has lead to a wide-spread use of computers to study and solve a variety of problems in algebraic geometry and related areas. One topic of particular interest in computational algebraic geometry, especially when numerical computations are utilized, is the ability to develop certificates of the computed result. Smale's α-theory [17] provides a method for certifying the quadratic convergence of Newton's method using data computed at one point. Since Newton's method is a foundational tool for numerically solving polynomial systems, the α-theoretic certificates provide a way to rigorously prove results following numerical computations. For example, the implementation of α-theory in `alphaCertified` [9,10] has been used to prove results in various applications, such as enumerative geometry [4,7,9] and potential energy landscapes arising

J. D. Hauenstein—Supported in part by NSF ACI-1460032 and Sloan Research Fellowship BR2014-110 TR14.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 34–50, 2017.
https://doi.org/10.1007/978-3-319-72453-9_3

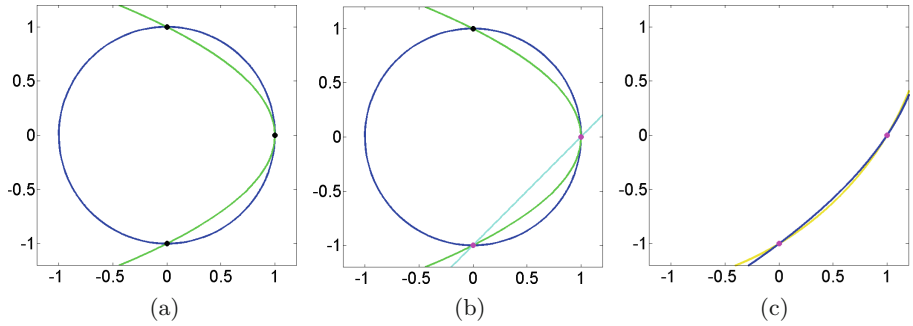


Fig. 1. Plots for (a) $f(x, y)$, (b) $G(x, y)$, and (c) $G_S(x, y)$ in (1)

in a physical or chemical system [13]. In these applications, which is common in many applications, one is interested in certifying the reality or nonreality of solutions. It was shown in [9] that certifying reality or nonreality is possible when the map corresponding to a Newton iteration is a real map, that is, maps real points to real points.

Two open problems related to α-theory are the ability to certify that an overdetermined system of analytic equations has a solution and to certify a singular solution for a square system of analytic equations. One can prove quadratic convergence of overdetermined Newton's method to critical points of the nonlinear least squares problem [5], some of which need not be solutions. By randomizing down to square systems, points which do not solve the overdetermined system can be certifiably identified [9]. For singular solutions, the behavior of Newton's method nearby can vary drastically (e.g., convergence, repulsion, and attracting cycles). Theorem 4 and Corollary 1 make progress towards these open problems via Newton-invariant subspaces.

To illustrate the results presented in Theorem 4 and Corollary 1 with Lemma 1, consider

$$f(x,y) = \begin{bmatrix} x^2+y^2-1 \\ x+y^2-1 \end{bmatrix}, \; G(x,y) = \begin{bmatrix} f(x,y) \\ x-y-1 \end{bmatrix}, \text{ and } G_S(x,y) = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \end{bmatrix} \cdot G(x,y) \tag{1}$$

which are plotted in Fig. 1. Example 1 shows that the set defined by $y = x - 1$ is Newton-invariant with respect to f. That is, if the input of Newton's method applied to f is a point on the line $y = x - 1$, the resulting point will also be on the line $y = x - 1$. Even though G_S is a randomized square system, G is overdetermined, and one of the two solutions of $G = 0$ is singular with respect to f, Theorem 4 and Corollary 1 together with Lemma 1 show that G_S and G can be used to prove the quadratic convergence of Newton's method to solutions of $f = 0$.

Newton-invariant sets can be considered as "side conditions." The algorithm **Certify** described in Sect. 3 certifiably decides if a point ξ which is a solution

of a square system $f = 0$ is contained in a Newton-invariant set V or in its complement $\mathbb{C}^n \setminus V$ using α-theory applied to a given numerical approximation of ξ. The "side conditions" could be defined via analytic equations, such as $y = x - 1$. Another naturally arising case is deciding "reality" of solutions in various coordinate systems. As mentioned above, the approach of [9] focuses on reality of solutions in Cartesian coordinates. When the map defined by a Newton iteration is a real map, the set of real points is a Newton-invariant subspace that is not defined by analytic equations. For Cartesian coordinates, Remark 1 shows that **Certify** reduces to the real certification approach of [9]. However, even though the two approaches may appear similar, the use of certifying "side conditions" as well as certifying "reality" in other coordinate systems show that this generalization is useful in a wide variety of applications. For example, consider a harmonic univariate polynomial $h(z)$ [12]. That is, $h(z) = p(z) + q(\text{conj}(z))$ where p and q are univariate polynomials and $\text{conj}(z)$ is the complex conjugate of z. One can compute the solutions of $h = 0$ by letting z and $\overline{z}$ be independent variables and solving the system

$$F(z, \overline{z}) = \begin{bmatrix} p(z) + q(\overline{z}) \\ \overline{p}(\overline{z}) + \overline{q}(z) \end{bmatrix} = 0$$

where $\overline{p}$ and $\overline{q}$ are univariate polynomials obtained by conjugating each coefficient of p and q, respectively. In particular, the solutions of $h = 0$ correspond to the solutions of $F = 0$ lying on the Newton-invariant set $\{(t, \text{conj}(t)) \mid t \in \mathbb{C}\}$. Such isotropic coordinates also arise naturally in algebraic kinematics [19].

The remainder of this section summarizes Smale's α-theory. Section 2 considers Newton invariant sets with Sect. 3 describing the algorithm **Certify**. The main theoretical results are presented in Sect. 4 with Sect. 5 describing the implementation in `alphaCertified` along with examples.

1.1 Smale's α-theory

For an analytic map $f : \mathbb{C}^n \to \mathbb{C}^n$, the map $N_f : \mathbb{C}^n \to \mathbb{C}^n$ defined by

$$N_f(x) := \begin{cases} x - Df(x)^{-1} f(x) & \text{if } Df(x) \text{ is invertible,} \\ x & \text{otherwise,} \end{cases}$$

is a Newton iteration of f at x where $Df(x)$ is the Jacobian matrix of f at x. With this definition, N_f is globally defined with the set of fixed points being

$$\{x \in \mathbb{C}^n \mid f(x) = 0 \text{ or rank } Df(x) < n\}.$$

Therefore, if $Df(x)$ is invertible, $N_f(x) = x$ if and only if $f(x) = 0$.

For each $k \geq 1$, define

$$N_f^k(x) := \underbrace{N_f \circ \cdots \circ N_f}_{k \text{ times}}(x).$$

A point $x \in \mathbb{C}^n$ is said to be an *approximate solution* of $f = 0$ if there is a point $\xi \in \mathbb{C}^n$ such that $f(\xi) = 0$ and

$$\|N_f^k(x) - \xi\| \leq \left(\frac{1}{2}\right)^{2^k - 1} \|x - \xi\| \tag{2}$$

for each $k \geq 1$ where $\|\cdot\|$ is the Euclidean norm on $\mathbb{C}^n$. In this case, the point ξ is called the *associated solution* to x and the sequence $\{N_f^k(x)\}_{k\geq 1}$ converges quadratically to ξ.

Smale's α-theory describes sufficient conditions using data computable from f and x for certifying that x is an approximate solution of $f = 0$. The algorithm presented in Sect. 3 will be based on the following theorem, which follows from results presented in [3, Chap. 8].

Theorem 1. *Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be analytic and $x, y \in \mathbb{C}^n$ such that $Df(x)$ and $Df(y)$ are invertible. Define*

$$\begin{aligned}
\alpha(f, x) &:= \beta(f, x) \cdot \gamma(f, x), \\
\beta(f, x) &:= \|x - N_f(x)\| = \|Df(x)^{-1} f(x)\|, \text{ and} \\
\gamma(f, x) &:= \sup_{k\geq 2} \left\| \tfrac{Df(x)^{-1} D^k f(x)}{k!} \right\|^{\frac{1}{k-1}}.
\end{aligned}$$

1. *If x is an approximate solution of $f = 0$ with associated solution ξ, then $N_f(x)$ is also an approximate solution with associated solution ξ and $\|x - \xi\| \leq 2\beta(f, x) = 2\|x - N_f(x)\|$.*
2. *If $4 \cdot \alpha(f, x) < 13 - 3\sqrt{17}$, then x is an approximate solution of $f = 0$.*
3. *If $\alpha(f, x) < 0.03$ and $\|x - y\| \cdot \gamma(f, x) < 0.05$, then x and y are approximate solutions of $f = 0$ with the same associated solution.*

The value $\beta(f, x)$ is called the *Newton residual*. In the definition of $\gamma(f, x)$, $D^k f(x)$ is the k^{th} derivative of f [11, Chap. 5]. That is, $D^k f(x)$ is a symmetric tensor that one may view as a linear map from $S^k\mathbb{C}^n$, the k-fold symmetric power of $\mathbb{C}^n$, to $\mathbb{C}^n$ whose entries are all of the partial derivatives of f of order k. When restricting to polynomial systems, $D^k f(x) = 0$ for all sufficiently large k so that $\gamma(f, x)$ is a maximum over finitely many terms. That is, $\gamma(f, x)$ could be computed algorithmically. However, due to the possibly large-scale nature of this computation, a commonly used upper bound for $\gamma(f, x)$ for a polynomial system f is described in [16]. A similar upper bound for polynomial-exponential systems is presented in [8].

2 Newton-Invariant Sets

A set $V \subset \mathbb{C}^n$ is *Newton-invariant* with respect to an analytic system $f : \mathbb{C}^n \to \mathbb{C}^n$ if

1. $N_f(x) \subset V$ for every $x \in V$ and

2. $\lim_{k\to\infty} N_f^k(x) \in V$ for every $x \in V$ such that $\lim_{k\to\infty} N_f^k(x)$ exists.

Clearly, if $N_f(x) \in \mathbb{R}^n$ for every $x \in \mathbb{R}^n$, then $\mathbb{R}^n$ is Newton-invariant with respect to f. Additionally, the set of solutions of $f = 0$ is also Newton-invariant. The following two examples show other cases of Newton-invariant sets.

Example 1. Let $f : \mathbb{C}^2 \to \mathbb{C}^2$ be the system defined in (1). Since f has real coefficients, N_f is a real map so that $V_1 := \mathbb{R}^2$ is trivially a Newton-invariant set for f. Consider the sets

$$V_2 := \{(0, y) \mid y \in \mathbb{C}\}, \quad V_3 := V_2 \cap \mathbb{R}^2, \quad V_4 := \{(1, y) \mid y \in \mathbb{C}\}, \quad V_5 := V_4 \cap \mathbb{R}^2,$$

$$V_6 := \{(x, x-1) \mid x \in \mathbb{C}\}, \quad V_7 := V_6 \cap \mathbb{R}^2, \quad V_8 := \{(x, 1-x) \mid x \in \mathbb{C}\}, \text{ and } V_9 := V_8 \cap \mathbb{R}^2.$$

One can show that $V_2, \ldots, V_9$ are also Newton-invariant sets for f as follows. Symbolically, $N_f(x, y) = (x + \Delta x, y + \Delta y)$ where

$$\Delta x = \frac{x(x-1)}{2x-1} \text{ and } \Delta y = \frac{y}{2} + \frac{(x-1)^2}{2y(2x-1)} \tag{3}$$

assuming that $x \neq 1/2$ and $y \neq 0$. For these special cases, $\Delta x = \Delta y = 0$ so they are not a concern when showing Newton-invariance. The Newton-invariance of $V_2, \ldots, V_5$ follows directly from the fact that $\Delta x = 0$ when either $x = 0$ and $x = 1$, and that N_f is a real map. If $y = x - 1$, it is easy to verify that $\Delta x = \Delta y$ which yields the Newton-invariance of V_6 and V_7. Finally, the Newton-invariance of V_8 and V_9 follows from the fact that $\Delta x = -\Delta y$ when $y = 1 - x$.

Example 2. The inverse kinematics problem of an RR dyad is the computation of the required angles θ_1 and θ_2 of the revolute joints needed to position the end effector at the point $(p_x, p_y) \in \mathbb{R}^2$ given that the RR dyad is anchored at $(0, 0)$ with fixed leg lengths $\ell_1 > 0$ and $\ell_2 > 0$. In short, this corresponds to solving the equations

$$\ell_1 \cos\theta_1 + \ell_2 \cos\theta_2 - p_x = \ell_1 \sin\theta_1 + \ell_2 \sin\theta_2 - p_y = 0. \tag{4}$$

Following a commonly used technique in algebraic kinematics [19], we will transform these equations into a polynomial system based on isotropic coordinates. Let $i = \sqrt{-1}$ and define

$$z_j := \cos\theta_j + i \cdot \sin\theta_j, \quad \overline{z}_j := \cos\theta_j - i \cdot \sin\theta_j, \text{ and } p = p_x + i \cdot p_y.$$

After substitution into (4), simplification, and addition of Pythagorean identities, the resulting polynomial system is

$$F(z_1, \overline{z}_1, z_2, \overline{z}_2) = \begin{bmatrix} \ell_1 z_1 + \ell_2 z_2 - p \\ \ell_1 \overline{z}_1 + \ell_2 \overline{z}_2 - \text{conj}(p) \\ z_1 \overline{z}_1 - 1 \\ z_2 \overline{z}_2 - 1 \end{bmatrix}$$

where conj() denotes complex conjugation. In the isotropic coordinates $(z_1, \overline{z}_1, z_2, \overline{z}_2)$, the corresponding set of "real" points is

$$V := \{(z_1, \text{conj}(z_1), z_2, \text{conj}(z_2)) \mid z_j \in \mathbb{C}\}.$$

Since each $\ell_j > 0$, it is easy to verify that V is Newton-invariant with respect to F.

2.1 Finding Newton-Invariant Sets

In Example 1, Newton-invariant sets where determined by performing a Newton iteration involving $f : \mathbb{C}^n \to \mathbb{C}^n$. That is, one first symbolically performs a Newton iteration for f to compute $\Delta x = Df(x)^{-1} f(x)$. Then, for example, the linear Newton-invariant spaces are found by computing matrices A and vectors b such that $Ax + b = 0$ and $A\Delta x = 0$. One may also parameterize the linear space and find the parameterizations which hold for x and $x + \Delta x$. The following reconsiders Example 1 to highlight this procedure followed by a polynomial system considered by Griewank and Osborne [6].

Example 3. Consider lines in $\mathbb{C}^2$ which are invariant with respect to (3). That is, we aim to find $(m_1, m_2) \in \mathbb{P}^1$ and $b \in \mathbb{C}$ such that $m_1 \Delta x = m_2 \Delta y$ whenever $m_2 y = m_1 x + b$.

If $m_2 = 0$, then we take $m_1 = -1$ and aim to find $b \in \mathbb{C}$ such that $\Delta x = 0$ whenever $x = b$. From (3), it is clear that $b = 0$ or $b = 1$. These lines correspond with $V_2, \ldots, V_5$ in Example 1.

If $m_2 \neq 0$, then we take $m_2 = 1$ and aim to find $m_1, b \in \mathbb{C}$ such that $\Delta y = m_1 \Delta x$ whenever $y = m_1 x + b$. Upon substitution and simplification, this requirement is equivalent to solving

$$m_1^2 + 2bm_1 + 1 = b^2 - 1 = 0$$

which yields $(m_1, b) = (1, -1)$ or $(-1, 1)$. These lines correspond with $V_6, \ldots, V_9$ in Example 1.

Example 4. Consider computing all linear Newton-invariant sets of a polynomial system first considered in [6], namely

$$G(x, y) = \begin{bmatrix} 29x^3/16 - 2xy \\ y - x^2 \end{bmatrix}. \tag{5}$$

For this system, which has a multiplicity 3 root at the origin, Griewank and Osborne showed that Newton's method diverges to infinity for almost all initial points. We have

$$\Delta x = \frac{3x^3}{32y - 23x^2} \text{ and } \Delta y = \frac{29x^4 - 55x^2 y + 32y^2}{32y - 23x^2}. \tag{6}$$

From (6), it is easy to verify that the vertical line $x = 0$ (over $\mathbb{C}$ and over $\mathbb{R}$) defines the only linear Newton-invariant set for G. We revisit this example in Sect. 5.3.

For larger polynomial systems f, it may be challenging to symbolically perform a Newton iteration for f, e.g., computing $\Delta x = Df(x)^{-1}f(x)$, thereby making it difficult to find all (linear) Newton-invariant sets for f. However, for particular applications, one often knows which Newton-invariant sets are of interest. Moreover, one can construct systems having a particular Newton-invariant set, as shown in the following.

Theorem 2. *Let n_1 and n_2 be positive integers with $n = n_1 + n_2$. Let $g : \mathbb{C}^{n_1} \to \mathbb{C}^n$ such that $g(0) = 0$, $A : \mathbb{C}^{n_1} \to \mathbb{C}^{n \times n_2}$, and $h : \mathbb{C}^{n_2} \to \mathbb{C}^{n_2}$ all be analytic. Then, $V := \{0\} \times \mathbb{C}^{n_2} \subset \mathbb{C}^n$ is a Newton-invariant set with respect to the square analytic system $F : \mathbb{C}^n \to \mathbb{C}^n$ defined by*

$$F(x, y) = g(x) + A(x) \cdot h(y).$$

Moreover, if N_h is a real map, then $V_{\mathbb{R}} := V \cap \mathbb{R}^n$ is Newton-invariant with respect to F.

Proof. Suppose that $y^* \in \mathbb{C}^{n_2}$ such that $DF(0, y^*)$ is invertible. Thus, $A(0) \cdot Dh(y^*)$ is an $n \times n_2$ matrix of rank n_2 so that $Dh(y^*)$ is invertible. It is easy to verify that $\Delta x = 0$ and $\Delta y = Dh(y^*)^{-1}h(y^*)$ is the unique solution of

$$DF(0, y^*) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = F(0, y^*)$$

showing that V is Newton-invariant with respect to F. The remaining statement follows immediately from the fact that Δy is real whenever N_h is a real map.

By using a change of coordinates, it follows that every linear subspace of $\mathbb{C}^n$ and $\mathbb{R}^n$ is a Newton-invariant set for some square system.

3 Certification Algorithm for Square Systems

Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be analytic and $V \subset \mathbb{C}^n$ be Newton-invariant with respect to f. Given an approximate solution $x \in \mathbb{C}^n$ of $f = 0$, this section develops an algorithm which certifiably decides if $\xi \in V$ or $\xi \in \mathbb{C}^n \setminus V$ where ξ is the associated solution of x. This algorithm depends on a function which measures the distance between a given point and V, say $\delta_V : \mathbb{C}^n \to \mathbb{R}$ where

$$\delta_V(z) = \inf_{v \in V} \|z - v\|. \tag{7}$$

For example, $\delta_{\mathbb{R}^n}(z) = \|z - \mathrm{conj}(z)\|/2$ with Remark 1 showing how the following algorithm generalizes the test for determining if $\xi \in \mathbb{R}^n$ proposed in [9] when N_f is a real map. Additionally, if computing $\delta_V(z)$ exactly is difficult, note that the following algorithm can be easily modified to use upper and lower bounds on $\delta_V(z)$ such that the upper bound limits to zero as z approaches V and the lower bound becomes positive as z limits to a solution ρ of $f = 0$ provided $\delta_V(\rho) > 0$.

The following procedure is shown to be a correct algorithm by Theorem 3.

Procedure $b = \textbf{Certify}(f, x, \delta_V)$
Input A square analytic system $f : \mathbb{C}^n \to \mathbb{C}^n$ such that $\gamma(f, \cdot)$ can be computed (or bounded above) algorithmically, a point $x \in \mathbb{C}^n$ which is an approximate solution of $f = 0$ with associated solution ξ such that $Df(\xi)^{-1}$ exists, and a function $\delta_V : \mathbb{C}^n \to \mathbb{R}$ defined by (7) for some Newton-invariant subspace V which can be computed algorithmically.
Output A boolean b which is `true` if $\xi \in V$ and `false` if $\xi \notin V$.
Begin
1. Compute $\beta := \beta(f, x)$, $\gamma := \gamma(f, x)$, and $\alpha := \beta \cdot \gamma$.
2. If $\delta_V(x) > 2\beta$, **Return** `false`.
3. If $\alpha < 0.03$ and $\delta_V(x) < 0.05\gamma^{-1}$, **Return** `true`.
4. Update $x := N_f(x)$ and go to Step 1.

Theorem 3. *Procedure* **Certify** *is an algorithm, i.e., terminates after finitely many steps, and develops a certificate of the correct answer.*

Proof. Consider the setup described in **Certify**. To prove the theorem, we will first show that if **Certify** returns in Step 2 or in Step 3, then the return value is correct. Afterwards, we will show that **Certify** must terminate in finitely many steps. Since each step in **Certify** is algorithmic, this shows that **Certify** is an algorithm.

Suppose that **Certify** returned through Step 2. For every $v \in V$, the triangle inequality and Item 1 of Theorem 1 yields

$$\delta_V(x) \leq \|x - v\| \leq \|x - \xi\| + \|\xi - v\| \leq 2\beta(f, x) + \|\xi - v\|.$$

Therefore,

$$0 < \delta_V(x) - 2\beta(f, x) \leq \inf_{v \in V} \|\xi - v\| = \delta_V(\xi)$$

yielding $\xi \notin V$ since $\delta_V(\xi) > 0$.

Similarly, suppose that **Certify** returned through Step 3. Then, since

$$\delta_V(x) = \inf_{v \in V} \|x - v\| < 0.05\gamma^{-1},$$

there must exist $v^* \in V$ such that $\|x - v^*\| < 0.05\gamma^{-1}$. By Item 3 of Theorem 1, both x and v^* are approximate solutions of $f = 0$ with the same associated solution ξ. Since $v^* \in V$ and V is Newton-invariant, it follows that $\xi \in V$.

To show termination of **Certify**, suppose that $\xi \notin V$. Define $\delta := \delta_V(\xi) > 0$ and consider

$$B(\xi, \delta/8) = \{y \in \mathbb{C}^n \mid \|y - \xi\| \leq \delta/8\}.$$

Since $N_f^k(x) \to \xi$ as $k \to \infty$, there exists some integer k_0 such that $N_f^k(x) \in B(\xi, \delta/8)$ for all $k \geq k_0$. It immediately follows from the triangle inequality that $\delta_V(N_f^{k_0}(x)) \geq 7\delta/8$ and $\beta(f, N_f^{k_0}(x)) = \|N_f^{k_0}(x) - N_f^{k_0+1}(x)\| \leq \delta/4$. Thus,

$$\delta_V(N_f^{k_0}(x)) \geq 7\delta/8 > 2\beta(f, N_f^{k_0}(x))$$

showing that Step 2 will force **Certify** to return after at most k_0 loops.

Similarly, suppose that $\xi \in V$. Then, since $Df(\xi)^{-1}$ exists, $\gamma(f,z)$ is bounded in a neighborhood W of ξ, say by B. Thus, there exists an integer k_0 such that $N_f^k(x) \in W$ for all $k \geq k_0$ so that $\gamma(f, N_f^k(x)) \leq B$ for all $k \geq k_0$. Since $\beta(f, N_f^k(x)) \to 0$ as $k \to \infty$, we know that $\alpha(f, N_f^k(x)) \to 0$ as $k \to \infty$. Hence, there must exist some integer k_1 such that $\alpha(f, N_f^k(x)) < 0.025$ for all $k \geq k_1$. Since $\xi \in V$, Item 1 of Theorem 1 yields

$$\delta_V(z) \leq \|z - \xi\| \leq 2\beta(f,z) = 2\alpha(f,z)\gamma(f,z)^{-1} < 0.05\gamma(f,z)^{-1}$$

where $z := N_f^{k_1}(x)$. Therefore, Step 3 will force **Certify** to return after at most k_1 loops.

Remark 1. When N_f is a real map, $\mathbb{R}^n$ is an Newton-invariant subspace with respect to f. For $z \in \mathbb{C}^n$, let $\pi_{\mathbb{R}}(z) \in \mathbb{R}^n$ be the real part of z, i.e., $\pi_{\mathbb{R}}(z) = (z + \text{conj}(z))/2$. Hence,

$$\delta_{\mathbb{R}^n}(z) = \|z - \text{conj}(z)\|/2 = \|z - \pi_{\mathbb{R}}(z)\|.$$

Thus, **Certify** reduces to the algorithm **CertifyRealSoln** described in [9] in this case.

4 Systems Constructed from Newton-Invariant Sets

In algorithm **Certify**, Newton iterations were performed on the square system and used to test if a solution was contained in a given Newton-invariant set or its complement, even if the Newton-invariant set was not defined by analytic equations (complex conjugation is not analytic). In this section, we investigate overdetermined systems constructed from a linear Newton-invariant set and randomized square subsystems. In particular, Theorem 4 and Corollary 1 show that if Newton's method applied to such systems quadratically converges, then the limit point is a solution of the original square system, even if it is singular with respect to the original system. That is, the additional equations could turn a singular solution of the square system into a nonsingular solution of an overdetermined and randomized square subsystem with certifiable quadratic convergence. See Sects. 5.3 and 5.4 for examples involving traditional benchmarks.

The statements of Theorem 4 and Corollary 1 rely upon the following two definitions. For an analytic system $f : \mathbb{C}^n \to \mathbb{C}^n$, define

$$\text{Sing}_f := \{x \in \mathbb{C}^n \mid Df(x) \text{ is not invertible}\}.$$

Overdetermined Newton's method for an analytic system $g : \mathbb{C}^n \to \mathbb{C}^N$ (i.e., $n < N$) is

$$N_g(x) := x - Dg(x)^\dagger g(x)$$

where $Dg(x)^\dagger$ is the Moore-Penrose pseudoinverse of $Dg(x)$.

Unlike square systems, the fixed points of N_g need not be solutions of $g = 0$ and the fixed points for which the Jacobian is full rank need not be attracting. For the former, consider

$$g(x) = \begin{bmatrix} x \\ x - 4 \end{bmatrix}.$$

Clearly, $g = 0$ has no solutions but $x = 2$ is a fixed point of N_g and minimizes $\|g\|_2$. For the latter, consider the system adapted from [5]:

$$h(x) = \begin{bmatrix} x \\ x^2 + 1 \end{bmatrix}.$$

Clearly, $h = 0$ has no solutions but N_h has a fixed point at $x = 0$. It is shown in [5] that $x = 0$ is a repulsive point for Newton's method near the origin.

From a certification viewpoint, one can use the α-theoretic approach of [5] to prove quadratic convergence to fixed points of N_g. The fixed points of N_g which do not solve $g = 0$ can be certifiably identified using randomization via the approach of [9]. The following provides an approach for certifiably showing that a given fixed point of N_g is indeed a solution of $g = 0$ when g is constructed via Newton-invariant sets. As mentioned above, this fixed point may be a singular solution of the original square system used to construct such an overdetermined system g.

Since linear Newton-invariant sets for a system are invariant under a linear change of coordinates, we simplify the presentation of our results based on systems having a coordinate subspace as a Newton-invariant set.

Lemma 1. *Let $0 < m < n$ and $f : \mathbb{C}^m \times \mathbb{C}^{n-m} \to \mathbb{C}^n$ be an analytic system such that $V := \{0\} \times \mathbb{C}^{n-m} \subset \mathbb{C}^n$ is Newton-invariant with respect to f and $V \not\subset \mathrm{Sing}_f$. Let $g(y) = f(0, y)$ and $G(x, y) = \{f(x, y), x\}$. If $z \in \mathbb{C}^{n-m}$ such that $(0, z) \in V \setminus \mathrm{Sing}_f$, $R \in \mathbb{C}^{(n-m)\times n}$, and $S \in \mathbb{C}^{n\times(n+m)}$ such that rank $Dg_R(y) = n-m$ and rank $DG_S(0, z) = n$ where $g_R(y) = R\cdot g(y)$ and $G_S(x, y) = S\cdot G(x, y)$, then $N_{G_S}(0, z) = N_G(0, z) = N_f(0, z) = (0, N_{g_R}(z)) = (0, N_g(z))$.*

Proof. Let $\Delta := Df(0, z)^{-1} \cdot f(0, z)$. Since $(0, z) \in V$, $\Delta_i = 0$ for $i = 1, \ldots, m$. Let $\Delta z \in \mathbb{C}^{n-m}$ such that $\Delta = (0, \Delta z)$. Since $Dg_R(z)$ and $DG_S(0, z)$ have full column rank, the same is true for $Dg(z)$ and $DG_S(0, z)$. Thus, the statement follows since

$$DG(0, z) \cdot \Delta = G(0, z), \quad DG_R(0, z) \cdot \Delta = G_R(0, z), \quad Dg(z) \cdot \Delta z = g(z), \quad Dg_R(z) \cdot \Delta z = g_R(z).$$

Following the notation of Lemma 1, for simplicity, the following relate the square system f, the overdetermined system g, and the randomized square subsystem g_R. These can be trivially extended via Lemma 1 to the overdetermined system G and randomized square system G_S.

Theorem 4. *Let $0 < m < n$ and $f : \mathbb{C}^m \times \mathbb{C}^{n-m} \to \mathbb{C}^n$ be an analytic system such that $V := \{0\} \times \mathbb{C}^{n-m} \subset \mathbb{C}^n$ is Newton-invariant with respect to f and $V \not\subset \mathrm{Sing}_f$. Let $g(y) = f(0, y)$. If $(0, z) \in V \setminus \mathrm{Sing}_f$ and $R \in \mathbb{C}^{(n-m)\times n}$ such that $\{N^k_{g_R}(z)\}_{k\geq 1}$ quadratically converges to $\xi \in \mathbb{C}^{n-m}$ with rank $Dg_R(\xi) = n - m$ where $g_R(y) = R \cdot g(y)$, then $g(\xi) = f(0, \xi) = 0$.*

Proof. Since g_R is a square system with $N_{g_R}(\xi) = \xi$ and rank $Dg_R(\xi) = n - m$, we know $g_R(\xi) = 0$, $\alpha(g_R, \xi) = 0$, and $\gamma(g_R, \xi) < \infty$. Thus, Theorem 1(3) shows that, for the ball $B \subset \mathbb{C}^{n-m}$ centered at ξ with radius $0.05/\gamma(g_R, \xi) > 0$, Newton's method for g_R starting at any point in B is an approximate solution of $g_R = 0$ with associated solution ξ. Define $W_f := \{y \mid (0, y) \in V \setminus \mathrm{Sing}_f\} \subset \mathbb{C}^{n-m}$. Since V is not contained in Sing_f, it follows that $B \cap W_f$ is dense in B. Therefore, we can construct $\{z_\ell\}_{\ell \geq 1} \subset B \cap W_f$ such that $0 < \|z_\ell - \xi\| < \ell^{-1}$. In particular, this construction yields rank $Df(0, z_\ell) = n$ and rank $Dg_R(z_\ell) = n - m$ for all $\ell \geq 1$.

For each $\ell \geq 1$, let $\Delta z_\ell := Dg_R(z_\ell)^{-1} g_R(z_\ell) = z_\ell - N_{g_R}(z_\ell)$. By Lemma 1, we know $(0, \Delta z_\ell) = (0, z_\ell) - N_f(0, z_\ell) = Df(0, z_\ell)^{-1} f(0, z_\ell)$. If we assume that $\|\Delta z_\ell\| \leq 2 \cdot \ell^{-1}$, then

$$\|g(z_\ell)\| = \|f(0, z_\ell)\| = \|Df(0, z_\ell) \cdot (0, \Delta z_\ell)\| \leq 2 \cdot \|Df(0, z_\ell)\| \cdot \ell^{-1}.$$

By continuity, $g(z_\ell) = f(0, z_\ell) \to g(\xi) = f(0, \xi)$ and $Df(0, z_\ell) \to Df(0, \xi)$. Since $Df(0, \xi)$ is an $n \times n$ matrix with complex entries, we know $\|Df(0, \xi)\| < \infty$. Taking limits, we have $\|g(\xi)\| = \|f(0, \xi)\| = 0$. Hence, $g(\xi) = f(0, \xi) = 0$.

Therefore, all that remains is to show $\|\Delta z_\ell\| \leq 2 \cdot \ell^{-1}$ for all $\ell \geq 1$. To reach a contradiction, we assume that $\ell \geq 1$ such that $\|\Delta z_\ell\| > 2 \cdot \ell^{-1}$. By construction, $\ell^{-1} > \|z_\ell - \xi\| > 0$ so that

$$\|z_\ell - N_{g_R}(z_\ell)\| = \|\Delta z_\ell\| > 2 \cdot \|z_\ell - \xi\| > 0.$$

The triangle inequality yields

$$\|z_\ell - \xi\| + \|N_{g_R}(z_\ell) - \xi\| \geq \|z_\ell - N_{g_R}(z_\ell)\| = \|\Delta z_\ell\| > 2 \cdot \|z_\ell - \xi\| > 0$$

providing $\|N_{g_R}(z_\ell) - \xi\| > \|z_\ell - \xi\| > 0$. However, since $z_\ell \in B$, i.e., z_ℓ is an approximate solution of $g_R = 0$ with associated solution ξ, (2) yields the impossible statement

$$\frac{1}{2}\|z_\ell - \xi\| \geq \|N_{g_R}(z_\ell) - \xi\| > \|z_\ell - \xi\| > 0.$$

Corollary 1. *Let $0 < m < n$ and $f : \mathbb{C}^m \times \mathbb{C}^{n-m} \to \mathbb{C}^n$ be an analytic system such that $V := \{0\} \times \mathbb{C}^{n-m} \subset \mathbb{C}^n$ is Newton-invariant with respect to f and $V \not\subset \mathrm{Sing}_f$. Suppose that $g(y) = f(0, y)$. If $(0, z) \in V \setminus \mathrm{Sing}_f$ such that $\{N_g^k(z)\}_{k \geq 1}$ quadratically converges to $\xi \in \mathbb{C}^{n-m}$ with rank $Df(0, N_g^k(z)) = n$ for all $k \geq 1$ and rank $Dg(\xi) = n - m$, then $g(\xi) = f(0, \xi) = 0$.*

Proof. Since rank $Dg(\xi) = n - m$, there is a Zariski open and dense $\mathcal{U} \subset \mathbb{C}^{(n-m) \times n}$ such that, for all $R \in \mathcal{U}$, rank $Dg_R(\xi) = n$ where $g_R(x) = R \cdot g(x)$. Fix $R \in \mathcal{U}$. By Theorem 1(3), Newton's method for g_R starting at any point in the ball B centered at ξ with radius $0.05/\gamma(g_R, \xi) > 0$ quadratically converges to ξ. Since $N_g^k(z) \to \xi$, let $k_0 \geq 1$ such that $\{N_g^k(z)\}_{k \geq k_0} \subset B$. Since Dg_R is full rank on B, Lemma 1 yields $N_g^k(z) = N_{g_R}^k(z)$ for all $k \geq k_0$. The statement now follows immediately from Theorem 4.

Example 5. Let $f : \mathbb{C}^2 \to \mathbb{C}^2$ be the polynomial system defined in (1). Example 1 showed the complex line $V_6 := \{(x, x-1) \mid x \in \mathbb{C}\}$, which is defined by $x-y-1 = 0$, is Newton-invariant with respect to f. Restricting to V_6, $f = 0$ has two solutions, namely $\xi_1 = (0, -1)$ and $\xi_2 = (1, 0)$. One can verify that $Df(\xi_1)$ is invertible while ξ_2 is a singular solution of $f = 0$. The overdetermined polynomial system

$$G(x, y) = \begin{bmatrix} x + y^2 - 1 \\ x^2 + y^2 - 1 \\ x - y - 1 \end{bmatrix}$$

has rank $DG(\xi_1) = \text{rank } DG(\xi_2) = 2$, i.e., ξ_1 is nonsingular with respect to both f and G, but ξ_2 is singular with respect to f and nonsingular with respect to G. Using `alphaCertified` [10] with the points $z_1 = (1/250, -249/250)$ and $z_2 = (251/250, 1/250)$, and square subsystem

$$G_S(x, y) = \begin{bmatrix} x + y^2 - 1 + 3(x - y - 1) \\ x^2 + y^2 - 1 + 2(x - y - 1) \end{bmatrix},$$

we know that $\{N_G^k(z_j)\}_{k\geq 1} = \{N_{G_S}^k(z_j)\}_{k\geq 1}$ quadratically converges for $j = 1, 2$. Theorem 4 and Corollary 1 together with Lemma 1 yield that the corresponding limit points, ξ_j, are indeed solutions of $f = 0$.

5 Implementation Details and Examples

Before demonstrating the developed techniques on several examples, we first briefly summarize its implementation in `alphaCertified` [10].

5.1 Implementation in `alphaCertified`

The software program `alphaCertified` can perform α-theoretic computations in exact rational or arbitrary precision floating point arithmetic. When rational computations are utilized, the internal computations are certifiable. The analytic system f must either be a polynomial system or a polynomial-exponential system and presented with constants that are rational complex numbers, i.e., in $\mathbb{Q}[i]$. The value of $\gamma(f, x)$ is bounded above using [16] or [8], respectively.

The algorithm **Certify** is implemented in version 1.3 of `alphaCertified` as follows. For a Newton-invariant set $V \subset \mathbb{C}^n$, the function δ_V defined by (7) is assumed to be of the form

$$\delta_V(z) = \|z - (P \cdot z_{\mathbb{R}} + i \cdot Q \cdot z_{\mathbb{I}} + r)\| \tag{8}$$

for $n \times n$ matrices P and Q and n vector r with rational complex entries where

$$z_{\mathbb{R}} = (z + \text{conj}(z))/2 \text{ and } z_{\mathbb{I}} = i \cdot (\text{conj}(z) - z)/2.$$

For example, if $V = \mathbb{R}^n$, then $P = I_n$, $Q = 0$, and $r = 0$ where I_n is the $n \times n$ identity matrix. Additionally, if $V = \{(x, \text{conj}(x)) \mid x \in \mathbb{C}\} \subset \mathbb{C}^2$, one can easily verify that

$$P = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad Q = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \text{ and } r = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

5.2 A Basic Example

Reconsider the system f defined in (1) with the real Newton-invariant sets (see Example 1)

$$V_1 := \mathbb{R}^2, \quad V_3 := \{(0, y) \mid y \in \mathbb{R}\}, \quad V_5 := \{(1, y) \mid y \in \mathbb{R}\},$$
$$V_7 := \{(x, x-1) \mid x \in \mathbb{R}\}, \text{ and } V_9 := \{(x, 1-x) \mid x \in \mathbb{R}\}.$$

It is easy to verify that each δ_{V_j} can be presented in the form (8). Let P_j, Q_j, and r_j be the corresponding elements. Since each $V_j \subset \mathbb{R}^2$, we have $Q_j = 0$. Additionally, since the origin is contained in V_1 and V_3, we also have $r_1 = r_3 = 0$. The remaining elements are:

$$P_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, P_3 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, P_5 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, P_7 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, P_9 = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},$$

$$r_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; r_7 = \begin{bmatrix} 1/2 \\ -1/2 \end{bmatrix}, \; r_9 = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}.$$

Since the singular solution $(1, 0)$ of $f = 0$ was considered in Example 5, we now consider the two nonsingular solutions, namely $(0, \pm 1)$. Clearly, both of $(0, \pm 1)$ lie in V_1 and V_3, with one in V_7 and the other in V_9. Algorithm **Certify** in `alphaCertified` using exact rational arithmetic promptly proves the proceeding statement starting with the approximations

$$(1/1502 - i/3203, 1256/1255 + i/1842) \text{ and } (-1/2934 + i/8472, -1483/1482 - i/2384).$$

5.3 An Example from Griewank and Osborne

Reconsider the polynomial system G from [6] defined in (5) for which the vertical line $x = 0$ is a Newton-invariant set. For any $y \neq 0$, $N_G(0, y) = (0, 0)$ so that Newton's method converges to the only solution of $G = 0$ in one iteration.

We now consider applying Newton's method to the point $P = (10^{-16}, 1)$. Figure 2 plots the Newton residual, i.e., β, for the first 200 iterations of Newton's method starting at P computed using `alphaCertified`. As suggested by this plot, Newton's method diverges to infinity. However, one can easily verify that for the system

$$H(x, y) = \begin{bmatrix} G(x, y) \\ x \end{bmatrix}$$

as well as for a randomization of H down to a square system, Newton's method starting at P immediately quadratically converges to the origin in stark contrast to the results of [6].

5.4 A System with Embedded Points

Consider the system defining the cyclic 4-roots [2], namely

$$F_4(x_1, x_2, x_3, x_4) = \begin{bmatrix} x_1 + x_2 + x_3 + x_4 \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1 \\ x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_1 + x_4x_1x_2 \\ x_1x_2x_3x_4 - 1 \end{bmatrix}.$$

The solution set defined by $F_4 = 0$ has two irreducible curves while the ideal generated by F_4 in $\mathbb{C}[x_1, \ldots, x_4]$ also has 8 embedded points. It is easy to verify that the line $\{(-t, -t, t, t) \mid t \in \mathbb{C}\}$ is Newton-invariant with respect to F_4 and contains 4 of the embedded points, namely when $t = \pm 1, \pm\sqrt{-1}$. For the overdetermined system G_4 constructed by appending the linear polynomials $x_1 - x_2$, $x_1 + x_3$, and $x_1 + x_4$ to the system F_4, each of these four embedded points are nonsingular solutions of $G_4 = 0$. For a general randomization of G_4, `Bertini` [1] computed numerical approximations of its 20 nonsingular solutions. Using the approach of [9], we are able to use `alphaCertified` to certifiably determine 16 of these solutions do not solve $F_4 = 0$. With Theorem 4, we can now use `alphaCertified` to certifiably show the other 4 solve $F_4 = 0$.

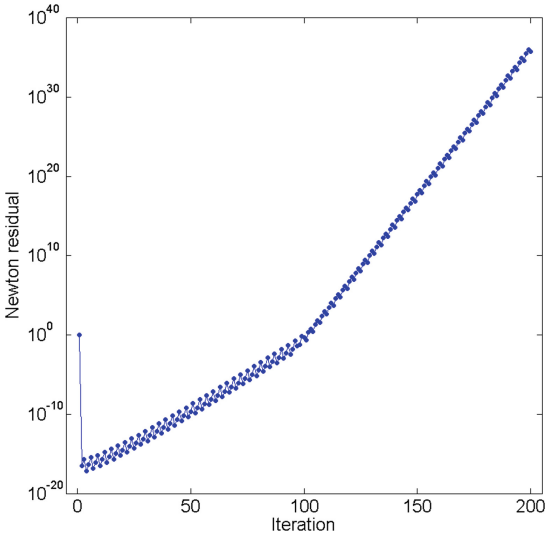


Fig. 2. Plot of the Newton residual for 200 iterations starting at P

5.5 Four-Bar Linkages Using Isotropic Coordinates

A general four-bar linkage moves in a one-dimensional motion curve when the joints are permitted to rotate. The nine-point path synthesis problem asks

to compute the one-dimensional motion curves of four-bar linkages that pass through nine given points. In [18], which showed there were 1442 motion curves passing through nine points in general position, the formulation of this problem used isotropic coordinates. Naturally, one may rewrite this system using Cartesian coordinates, which was used in the formulation of the problem in [15] and certification of real solutions in [9]. However, with **Certify**, one can certify directly using the isotropic formulation.

Let $\mathcal{P} = \{P_0, \ldots, P_8\} \subset \mathbb{C}^2$ be a collection of nine points written using isotropic coordinates. The polynomial system $f_{\mathcal{P}} : \mathbb{C}^{12} \to \mathbb{C}^{12}$ corresponding to the isotropic formulation of the nine-point path synthesis problem derived in [18] depends upon the variables

$$\{x, \overline{x}, a, \overline{a}, n, \overline{n}, y, \overline{y}, b, \overline{b}, m, \overline{m}\}$$

and is constructed as follows. The first four polynomials are

$$f_1 = n - a\overline{x}, \quad f_2 = \overline{n} - \overline{a}x, \quad f_3 = m - b\overline{y}, \text{ and } f_4 = \overline{m} - \overline{b}y.$$

The remaining eight polynomials arise from the displacement from P_0 to the other points P_j. Define $Q_j := (\delta_j, \overline{\delta}_j) = P_j - P_0$, which is written via isotropic coordinates. Then, for $j = 1, \ldots, 8$,

$$f_{4+j} = \gamma_j\overline{\gamma}_j + \gamma_j\gamma_j^0 + \overline{\gamma}_j\gamma_j^0$$

where

$$\gamma_j = q_j^x r_j^y - q_j^y r_j^x, \quad \overline{\gamma}_j = r_j^x p_j^y - r_j^y p_j^x, \quad \gamma_j^0 = p_j^x q_j^y - p_j^y q_j^x$$

and

$$\begin{array}{lll} p_j^x = \overline{n} - \overline{\delta}_j x, & q_j^x = n - \delta_j\overline{x}, & r_j^x = \delta_j(\overline{a} - \overline{x}) + \overline{\delta}_j(a - x) - \delta_j\overline{\delta}_j, \\ p_j^y = \overline{m} - \overline{\delta}_j y, & q_j^y = m - \delta_j\overline{y}, & r_j^y = \delta_j(\overline{b} - \overline{y}) + \overline{\delta}_j(b - y) - \delta_j\overline{\delta}_j. \end{array}$$

When $\mathcal{P}$ consists of points in general position, there is a six-to-one map from the solution set of $f_{\mathcal{P}} = 0$ to four-bar motion curves which pass through the points $\mathcal{P}$ arising from a two-fold symmetry and Roberts cognates. Moreover, when $\mathcal{P}$ consists of 9 points that are "real" in isotropic coordinates, then $\overline{\delta}_j = \text{conj}(\delta_j)$ and

$$V := \{(x, \text{conj}(x), a, \text{conj}(a), n, \text{conj}(n), y, \text{conj}(y), b, \text{conj}(b), m, \text{conj}(m)) \mid x, a, n, y, b, m \in \mathbb{C}\}$$

is Newton-invariant with respect to $f_{\mathcal{P}}$.

As a demonstration of the algorithm **Certify**, we certify the solution to two sets of real points. The first, called Problem 3 in Table 2 of [18], was also considered in [9] using Cartesian coordinates. The corresponding δ_j are

$$\begin{array}{l} \delta_1 = 0.27 + 0.1i, \quad \delta_2 = 0.55 + 0.7i, \delta_3 = 0.95 + i, \quad \delta_4 = 1.15 + 1.3i, \\ \delta_5 = 0.85 + 1.48i, \delta_6 = 0.45 + 1.4i, \delta_7 = -0.05 + i, \delta_8 = -0.23 + 0.4i \end{array}$$

with $\overline{\delta}_j = \text{conj}(\delta_j)$. We used **Certify** implemented in `alphaCertified` to certify the approximations of the solutions obtained by `Bertini` [1]. Confirming

previous computations in [9,18], this showed that 64 of the 1442 motion curves through the corresponding nine points were real.

The second set of real points is modeled after Problem 4 in Table 2 of [18] which took points on the ellipse $x^2 + y^2/4 = 1$. Since that collection of nine points was contained in the discriminant locus, we took a collection of nine points on this ellipse and perturbed them. For example, consider the following δ_j, with $\overline{\delta}_j = \text{conj}(\delta_j)$, constructing in this fashion:

$$\delta_1 = 0.25 + 1.33i,\ \delta_2 = 0.5 + 1.74i,\ \delta_3 = 0.75 + 1.93i,\ \delta_4 = 1 + 2.01i,$$
$$\delta_5 = 1.25 + 1.95i,\ \delta_6 = 1.5 + 1.73i,\ \delta_7 = 1.75 + 1.33i,\ \delta_8 = 2 - 0.007i.$$

After using `Bertini` to generate approximations to the solutions, **Certify** showed that 51 of the 1442 motion curves through the corresponding nine points were real.

6 Discussion and Summary

Newton-invariant sets naturally arise when considering "real" solutions in other coordinate systems. They could also arise in other situations, such as "side conditions" for solution sets. Theorem 4 and Corollary 1 provide conditions in which the limit of Newton's method applied to an overdetermined system or a randomized square subsystem converges to a true solution. This article also described the algorithm **Certify** which, from an approximation of a solution, can certifiably determine if the corresponding solution is contained in a particular Newton-invariant set or its complement. This approach adds to the certifiable toolbox of methods that can be applied to various problems in computational algebraic geometry.

The algorithm **Certify** is implemented in `alphaCertified` using both exact rational and arbitrary precision floating point arithmetic. All computations are completely rigorous when using rational arithmetic. If floating point arithmetic is used, the current implementation does not fully control roundoff errors. One could, for example, use interval arithmetic [14] to bound the errors and produce certifiable computations in this case.

Acknowledgments. The author would like to thank Charles Wampler for helpful discussions related to using isotropic coordinates in kinematics.

References

1. Bates, D.J., Hauenstein, J.D., Sommese, A.J., Wampler, C.W.: Bertini: Software for Numerical Algebraic Geometry. bertini.nd.edu
2. Björck, G., Fröberg, R.: A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots. J. Symbolic Comput. **12**(3), 329–336 (1991)
3. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and Real Computation. Springer, New York (1998). https://doi.org/10.1007/978-1-4612-0701-6

4. Bozóki, S., Lee, T.-L., Rónyai, L.: Seven mutually touching infinite cylinders. Comput. Geom. **48**(2), 87–93 (2015)
5. Dedieu, J.-P., Shub, M.: Newton's method for overdetermined systems of equations. Math. Comput. **69**(231), 1099–1115 (2000)
6. Griewank, A., Osborne, M.R.: Analysis of Newton's method at irregular singularities. SIAM J. Numer. Anal. **20**(4), 747–773 (1983)
7. Griffin, Z.A., Hauenstein, J.D.: Real solutions to systems of polynomial equations and parameter continuation. Adv. Geom. **15**(2), 173–187 (2015)
8. Hauenstein, J.D., Levandovskyy, V.: Certifying solutions to square systems of polynomial-exponential equations. J. Symbolic Comput. **79**(3), 575–593 (2017)
9. Hauenstein, J.D., Sottile, F.: Algorithm 921: alphaCertified: certifying solutions to polynomial systems. ACM TOMS **38**(4), 28 (2012)
10. Hauenstein J.D., Sottile, F.: alphaCertified: software for certifying solutions to polynomial systems. www.math.tamu.edu/~sottile/research/stories/alphaCertified
11. Lang, S.: Real Analysis, 2nd edn. Addison-Wesley Publishing Company Advanced Book Program, Reading (1983)
12. Li, W.V., Wei, A.: On the expected number of zeros of a random harmnoic polynomial. Proc. Am. Math. Soc. **137**(1), 195–204 (2009)
13. Mehta, D., Hauenstein, J.D., Wales, D.J.: Certifying the potential energy landscape. J. Chem. Phys. **138**(17), 171101 (2013)
14. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. SIAM, Philadelphia (2009)
15. Roth, B., Freudenstein, F.: Synthesis of path-generating mechanisms by numerical methods. ASME J. Eng. Ind. **85**(3), 298–306 (1963)
16. Shub, M., Smale, S.: Complexity of Bézout's theorem. I. Geometric aspects. J. Am. Math. Soc. **6**(2), 459–501 (1993)
17. Smale, S.: Newton's method estimates from data at one point. In: Ewing, R.E., Gross, K.I., Martin, C.F. (eds.) The Merging of Disciplines: New Directions in Pure, Applied, and Computational Mathematics (Laramie, Wyo 1985), pp. 185–196. Springer, New York (1986). https://doi.org/10.1007/978-1-4612-4984-9_13
18. Wampler, C.W., Morgan, A.P., Sommese, A.J.: Complete solution of the nine-point path synthesis problem for four-bar linkages. ASME J. Mech. Des. **114**, 153–159 (1992)
19. Wampler, C.W., Sommese, A.J.: Numerical algebraic geometry and algebraic kinematics. Acta Numer. **20**, 469–567 (2011)

Decomposition of Low Rank Multi-symmetric Tensor

Jouhayna Harmouch[1,2(✉)], Bernard Mourrain[2(✉)], and Houssam Khalil[1]

[1] Laboratory of Mathematics and Its Applications LaMA-Lebanon, Lebanese University, Beirut, Lebanon
houssam.khalil@ul.edu.lb
[2] UCA, Inria, AROMATH, Sophia Antipolis, France
{jouhayna.harmouch,Bernard.Mourrain}@inria.fr

Abstract. We study the decomposition of a multi-symmetric tensor T as a sum of powers of product of linear forms in correlation with the decomposition of its dual T^* as a weighted sum of evaluations. We use the properties of the associated Artinian Gorenstein Algebra A_τ to compute the decomposition of its dual T^* which is defined via a formal power series τ. We use the low rank decomposition of the Hankel operator H_τ associated to the symbol τ into a sum of indecomposable operators of low rank. A basis of A_τ is chosen such that the multiplication by some variables is possible. We compute the sub-coordinates of the evaluation points and their weights using the eigen-structure of multiplication matrices. The new algorithm that we propose works for small rank. We give a theoretical generalized approach of the method in n dimensional space. We show a numerical example of the decomposition of a multi-linear tensor of rank 3 in 3 dimensional space.

1 Introduction

The decomposition of symmetric and multi-symmetric tensors has many applications in engineering disciplines such that signal processing [11], scientific data analysis [9,18], statistics [17], in bioinformatics and spectroscopy [5], in neuroscience, in phylogenetic.... For instance, the study of symmetric tensor decomposition gives an idea about the geometric structure of intersecting fibers in human brain using the Fibers Orientation Fibers Function described in [12], ou [8,19]. The decomposition of multi symmetric tensors of small rank appear in several other contexts, for learning latent variable models which are algebraic statistics models. This is the case for the analysis of phylogenetic trees model described in [15] or for the analysis of contents of web pages model described in [1]. Here, the mixture model is a collection of all non-negative probability tensors of low rank.

The tensor decomposition problem is also very interesting from an algebraic geometric point of view [10]. Important efforts have been developed over the last decades to better understand the theoretical aspects, as well as the algorithmic aspects of this difficult problem. Some of the well-known decomposition

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 51–66, 2017.
https://doi.org/10.1007/978-3-319-72453-9_4

methods use local optimization techniques such Alternate Least Square, Gradient Descents, Quasi-Newton, ... to minimize the error between the tensor and its decomposition. Some other approaches exploit the algebraic structure associated to the tensor decomposition [3,4]. Homotopy techniques have also been used recently to compute such decomposition [2].

In this paper, we describe a direct method for the decomposition of multi-symmetric tensors, based on simple linear algebra tools. The decomposition algorithm applies to tensors of low enough rank. We follow the approach in [3] but directly apply numerically stable linear algebra tools on submatrices of the Hankel matrices to recover the decomposition. In particular, we show how to recover directly the points and weights from eigenvectors of multiplication operators of the quotient algebra associated to the decomposition. The algorithm does not require the solution of polynomial equations. The proposed method extends the techniques of [16] to more general tensors and to tensors of higher rank. It is closely connected to the multivariate Prony method investigated in [14] and to the structured low rank decomposition of Hankel matrix [7].

A multi-linear tensor is in correspondence with a multilinear map from a product of vector spaces to the coefficient field. A tensor symmetric tensor is a tensor whose components stay invariant by any permutation of indices. In the following, we study the general class of multi symmetric tensor decomposition problem, which contains these two classes. We show the correlation between the dual of a tensor, formal power series and then the Hankel matrices associated to them. We use the singular value decomposition of Hankel matrices to compute the decomposition of a tensor of low rank. We exploit the properties of Artinian Gorenstein Algebra to find out some multiplication matrices which help to know the eigen-structure of points associated to linear forms and their weights. We slice variables into bunches of sub-variables and we adapt the description of Artinian Gorenstein Algebra to this case. We adapt the method of decomposition of Hankel matrices of low rank described in [7] to a decomposition of multi linear tensors method which is based on the decomposition of a formal power series as a weighted sum of exponential described in [14]. The computation of multiplication matrices depend on the dimension of tensor, and the number of given moments or coefficients. We describe the algorithm in 3 dimensional space and we give its numerical implementation using MAPLE. This description gives an idea about the constraints and difficulties of the problem in n dimensional space. We show a numerical example of the decomposition of a tensor of rank 3 with order one in each bunch of 3 variables in 3 dimensional space.

Contributions. We study the decomposition of multi-linear tensor T as a sum of product of powers of linear forms in correlation with the decomposition of its dual T^* as a weighted sum of evaluations. T^* is defined via a formal power series τ. We exploit the structure of the quotient algebra A_τ of the ring of multivariate polynomials in bunches of sub-variables by the kernel of the Hankel operator H_τ associated to τ. We choose two bases A_1 and A_2 of monomials such that all given moments of the tensor T appear in the matrix H_τ associated to T in the bases A_1 and A_2 and we substitute x_0 by one. We compute the Singular Value

Decomposition of the Hankel matrix associated to a chosen truncated bases of A_1 and A_2 such that the multiplication of the matrix by one fixed variable is well defined. We exploit the eigen-structure properties of multiplication operators to compute the sub-coordinates of points and their corresponding weights. We show the constraints which arise from the computation of all multiplication matrices in higher dimension spaces. We propose a new algorithm to compute the sub-coordinates of points using the eigenvalues of multiplication matrices and their transpose. We deduce weights from eigenvectors of a linear combination of multiplication matrices. This method is an adaptation of Structured Low Rank Decomposition of Multivariate Hankel Matrices method proposed in [7] which is the generalization of Prony method. We give a numerical interpretation of the decomposition of a multi-linear tensor of low rank in 3 dimensional space.

Structure of the paper. In the following section, we recall the definition of multi symmetric tensors of rank r and the affine decomposition theory of them. In Sect. 2, we recall some important properties of Artinian Gorenstein Algebra that we adapt to solve the dual decomposition problem which is resumed by the computation of points and their weights. In Sect. 3, we give a theoretical approach of the multi linear symmetric decomposition problem and a new algorithm to solve the decomposition problem in 3 dimensional space. In Sect. 4, we give an implementation of our algorithm for one example using MAPLE and we interpret the results.

2 Partial Symmetric Tensor Decomposition Problem

In this section we give the definition of a multi-symmetric tensor as a multi-homogeneous polynomial of a different positive degree at each collection of variables. This polynomial can be defined as well as multi symmetric array of coefficients. In the opposite, for a multi symmetric array of coefficients we can define a multi-homogeneous polynomial and then deshomogenize it. We recall the definition of minimal affine decomposition of a multi symmetric tensor as weighted sum of product of power of linear forms. We show the relationship between the dual of deshomogenized tensor and the formal power series associated to it using the apolar product. Then, after scaling by the linear form of the decomposition and multiplying the weights by the scaling factor we deduce by linearity that the dual of the Tensor can be decomposed as a weighted sum of evaluations.

Definition 1. *Let $(E_j)_{1\leq j\leq k}$ be a family of n_j+1 dimensional vector spaces, each one of them is of basis $\mathbf{x}_j$ such that $E_j = \langle \mathbf{x}_j \rangle = \langle x_j, \ldots, x_{j,n_j} \rangle$.*

Definition 2. *$\mathcal{S}^{\delta_j}(E_j)$ is the vector space of homogeneous polynomials in the variables $\mathbf{x}_j$ of degree δ_j.*

Definition 3. *$\mathcal{S}^{\delta_1}(E_1) \otimes \mathcal{S}^{\delta_2}(E_2) \otimes \ldots \otimes \mathcal{S}^{\delta_k}(E_k)$ is the vector space of multi-homogeneous polynomials of degree δ_j in each subset of variables $\mathbf{x}_j$ for $j = 1, \ldots, k$, an element $[T]$ of this vector space is called a multi symmetric tensor. It is denoted hereafter as $\mathcal{S}^{\delta}(E)$.*

Definition 4. *A multi symmetric tensor of* $\mathcal{S}^{\delta_1}(E_1) \otimes \mathcal{S}^{\delta_2}(E_2) \otimes \ldots \otimes \mathcal{S}^{\delta_k}(E_k)$ *can be interpreted as a multi symmetric array of coefficients* $[T] = [t_{\alpha_1',\alpha_2',\ldots,\alpha_k'}]_{\substack{|\alpha_j'|=\delta_j \\ \alpha_j' \in \mathbb{N}^{n_j+1}}}$ *such that each* $\alpha_j' = (\alpha_{j,p_j}')_{0 \leq p_j \leq n_j}$ *is a multi-index for* $1 \leq j \leq k$.

For $\alpha \in \mathbb{N}^n$ with $|\alpha| \leq \delta$, we denote $\bar{\alpha} = (\delta - |\alpha|, \alpha_1, \ldots, \alpha_n)$. The multi symmetric tensor is defined as $[T] = [t_{\bar{\alpha}_1,\bar{\alpha}_1,\ldots,\bar{\alpha}_k}]_{\substack{|\alpha_j| \leq \delta_j \\ \alpha_j \in \mathbb{N}^{n_j}}}$.

Such tensor is identified with the multi-homogeneous polynomial

$$T(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k) = \sum_{\substack{|\bar{\alpha}_j|=\delta_j \\ \bar{\alpha}_j \in \mathbb{N}^{n_j+1}}} t_{\bar{\alpha}_1,\bar{\alpha}_2,\ldots,\bar{\alpha}_k} (\mathbf{x}_1)^{\bar{\alpha}_1} (\mathbf{x}_2)^{\bar{\alpha}_2} \ldots (\mathbf{x}_k)^{\bar{\alpha}_k}$$

If we let $x_j = 1$ for $j = 1, \ldots, k$ we get

$$\underline{T}(\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \ldots, \underline{\mathbf{x}}_k) = \sum_{\substack{|\alpha_j| \leq \delta_j \\ \alpha_j \in \mathbb{N}^{n_j}}} t_{\alpha_1,\alpha_2,\ldots,\alpha_k} (\underline{\mathbf{x}}_1)^{\alpha_1} (\underline{\mathbf{x}}_2)^{\alpha_2} \ldots (\underline{\mathbf{x}}_k)^{\alpha_k}$$

where $\underline{\mathbf{x}}_j = (x_{j,1}, \ldots, x_{j,n_j})$ for $j = 1, \ldots, k$ because of $(\underline{\mathbf{x}}_j)^{\alpha_j} = (\mathbf{x}_j)^{\bar{\alpha}_j}$ for $j = 1, \ldots, k$.

A multilinear tensor is defined when $|\bar{\alpha}_j| = \delta_j = 1$ for $j = 1, \ldots, k$, then by abuse of notation we obtain $\bar{\alpha}_j[i_j] = 1$ for some $0 \leq i_j \leq n_j$ and 0 elsewhere, so that the multi symmetric array associated to that tensor is defined as $[T] = [t_{i_1,i_2,\ldots,i_k}]_{\substack{0 \leq i_j \leq n_j \\ 1 \leq j \leq k}}$

Given e_j basis of E_j for $j = 1, \ldots, k$, the tensor $[T]$ in the basis $e_1 \otimes e_2 \otimes \ldots \otimes e_k$ is equal to $T = \sum_{\substack{0 \leq i_1 \leq n_1 \\ 0 \leq i_2 \leq n_2 \\ \vdots \\ 0 \leq i_k \leq n_k}} t_{i_1,i_2,\ldots,i_k} e_{1,i_1} \otimes e_{2,i_2} \otimes \ldots \otimes e_{k,i_k}$, such a tensor can be identified with the multi-homogeneous polynomial $T(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k) = \sum_{\substack{0 \leq i_1 \leq n_1 \\ 0 \leq i_2 \leq n_2 \\ \vdots \\ 0 \leq i_k \leq n_k}} t_{i_1,i_2,\ldots,i_k} x_{1,i_1} x_{2,i_2} \ldots x_{k,i_k}$ because of $(\mathbf{x}_j)^{\bar{\alpha}_j} = x_{j,i_j}$ for some $0 \leq i_j \leq n_j$ and for all $1 \leq j \leq k$.

The dual of the tensor is $T^*(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k) = \sum_{\substack{0 \leq i_1 \leq n_1 \\ 0 \leq i_2 \leq n_2 \\ \vdots \\ 0 \leq i_k \leq n_k}} t_{i_1,i_2,\ldots,i_k} y_{1,i_1} y_{2,i_2} \ldots y_{k,i_k}$ because of $(\mathbf{y}_j)^{\bar{\alpha}_j} = y_{j,i_j}$ for some $0 \leq i_j \leq n_j$ and for all $1 \leq j \leq k$.

We denote $R_{\delta_1,\delta_2,\ldots,\delta_k}$ the space obtained by the deshomogeneisation of elements in $\mathcal{S}^{\delta}(E)$ by setting $x_j = 1$ for $j = 1, \ldots, k$ where $R = \mathbb{C}[\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \ldots, \underline{\mathbf{x}}_k]$ is the space of polynomials in the variables $\underline{\mathbf{x}}_j = (x_{j,1}, \ldots, x_{j,n_j})$ for $j = 1, \ldots, k$

Definition 5. *The tensor decomposition problem of* $T(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ *is the decomposition of* T *as a sum of product of power of linear forms such that*

$T(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k) = \sum_{p=1}^{r} \omega_p \mathbf{u}_{p,1}^{\delta_1}(\mathbf{x}_1)\mathbf{u}_{p,2}^{\delta_2}(\mathbf{x}_2)\ldots\mathbf{u}_{i,k}^{\delta_k}(\mathbf{x}_k)$ *where* $\mathbf{u}_{p,j}(\mathbf{x}_j) = u_{p,j}x_j + u_{p,j,1}x_{j,1} + \ldots + u_{p,j,n_j}x_{j,n_j}$ *and*

$$\mathbf{u}_p = (u_{p,j,p_j})_{\substack{0\le p_j\le n_j \\ 1\le j\le k}} = (u_{p,1}, u_{p,1,1}, \ldots, u_{p,1,n_1}, u_{p,2}, u_{p,2,1}, \ldots, u_{p,2,n_2}, \ldots\ldots,$$
$$u_{p,k}, u_{p,k,1}, \ldots, u_{p,k,n_k}) \in \mathbb{C}^{\sum_{j=1}^{k}(n_j+1)}$$

is the coefficient vector associated to the linear forms $\mathbf{u}_{p,j}(\mathbf{x}_j)$ *in the basis* $\mathbf{x}_j$ *for* $j = 1, \ldots, k$.

Definition 6. *The minimal number of terms in a decomposition of* $T(\mathbf{x})$ *is called the rank of* T.

We say that $T(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ has an affine minimal decomposition of the previous form if $u_{p,j} \neq 0$ for $p = 1, \ldots, r$ and $j = 1, \ldots, k$ where r is the rank of T.

Definition 7. *For* $T = (t_{\alpha_1,\alpha_2,\ldots,\alpha_k})_{\substack{|\alpha_j|\le\delta_j \\ \alpha_j\in\mathbb{N}^{n_j}}} \in \mathcal{S}^{\delta}(E)$ *we denote* $\tau_{\alpha_1,\alpha_2,\ldots,\alpha_k}(T) = \tau_{\alpha_1,\alpha_2,\ldots,\alpha_k} = t_{\alpha_1,\alpha_2,\ldots,\alpha_k}\binom{\delta_1}{\alpha_1}^{-1}\binom{\delta_2}{\alpha_2}^{-1}\ldots\binom{\delta_k}{\alpha_k}^{-1}$. *The dual of the tensor* $T(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k) \in \mathcal{S}^{\delta}(E)$ *is defined via the formal power series as* $\tau(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k) = T^*(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k) = \sum_{\substack{|\alpha_j|\le\delta_j \\ \alpha_j\in\mathbb{N}^{n_j}}} \tau_{\alpha_1,\alpha_2,\ldots,\alpha_k}\frac{(\mathbf{y}_1)^{\bar{\alpha}_1}}{\bar{\alpha}_1!}\frac{(\mathbf{y}_2)^{\bar{\alpha}_2}}{\bar{\alpha}_2!}\cdots \frac{(\mathbf{y}_k)^{\bar{\alpha}_k}}{\bar{\alpha}_k!}$ *where* $(\mathbf{y}_j)^{\bar{\alpha}_j} = (y_j, y_{j,1}, \ldots, y_{j,n_j})^{(\alpha_j,\alpha_{j,1},\ldots,\alpha_{j,n_j})} = \prod_{p_j=0}^{n_j}(y_{j,p_j})^{\alpha_{j,p_j}}$ *for* $j = 1, \ldots, k$

Definition 8. *For a polynomial* $p \in R$ *and a formal power series* $\tau \in R^*$, *we define the multiplication operator* $*$ *such that*

$$p * \tau : R \to \mathbb{C}$$
$$q \mapsto \tau(p.q)$$

Definition 9. *Let* $T_1(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ *and* $T_2(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ *be two tensors of* $\mathcal{S}^{\delta}(E)$. *The apolar product of* $\underline{T_1}(\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \ldots, \underline{\mathbf{x}}_k)$ *and* $\underline{T_2}(\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \ldots, \underline{\mathbf{x}}_k)$ *is defined as*

$$\langle \underline{T_1}(\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \ldots, \underline{\mathbf{x}}_k), \underline{T_2}(\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \ldots, \underline{\mathbf{x}}_k)\rangle = \sum_{\substack{|\alpha_j|\le\delta_j \\ \alpha_j\in\mathbb{N}^{n_j}}} \tau^{(1)}_{\alpha_1,\alpha_2,\ldots,\alpha_k}\bar{\tau}^{(2)}_{\alpha_1,\alpha_2,\ldots,\alpha_k}\binom{\delta}{\alpha}$$

where $\binom{\delta}{\alpha} = \binom{\delta_1}{\alpha_1}\binom{\delta_2}{\alpha_2}\ldots\binom{\delta_k}{\alpha^k}$.

Definition 10. *The dual operator of a tensor is defined as*

$$T^* : (R_{\delta_1,\delta_2,\ldots,\delta_k}) \to (R_{\delta_1,\delta_2,\ldots,\delta_k})^* \quad (1)$$
$$\underline{T_2} \mapsto T^*(\underline{T_2}) = \langle \underline{T}(\underline{\mathbf{x}}), \underline{T_2}(\underline{\mathbf{x}})\rangle \quad (2)$$

Lemma 1. *By a generic change of coordinates in each* E_j, *we may assume that* $u_{p,j} \neq 0$ *and that* T *has an affine decomposition. Then by scaling* $\mathbf{u}_p(\mathbf{x})$ *and multiplying* ω_p *by the* $d^{\boldsymbol{th}}$ *power of the scaling factor we may assume that* $u_{p,j} = 1$ *for* $p = 1, \ldots, r$ *and* $j = 1, \ldots, k$. *Thus the polynomial* $\underline{T}(\underline{\mathbf{x}}) = \sum_{p=1}^{r}\omega'_p{\mathbf{u}'_p}^{\delta}(\underline{\mathbf{x}}) = \sum_{p=1}^{r}\omega'_i{\mathbf{u}'_{p,1}}^{\delta_1}(\underline{\mathbf{x}}_1){\mathbf{u}'_{p,2}}^{\delta_2}(\underline{\mathbf{x}}_2)\ldots{\mathbf{u}'_{p,k}}^{\delta_k}(\underline{\mathbf{x}}_k)$.

Proposition 1. *The dual of the product of powers of linear forms* $\mathbf{u}_1^{\delta_1}\mathbf{u}_2^{\delta_2}\dots\mathbf{u}_k^{\delta_k}$ *is the evaluation* $\mathbf{e}_\mathbf{u}$ *at* $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k)$.

Proof. For $T = \mathbf{u}_1^{\delta_1}\mathbf{u}_2^{\delta_2}\dots\mathbf{u}_k^{\delta_k}$ and any $T' \in R_{\delta_1,\delta_2,\dots,\delta_k}$, we check that $\langle \underline{T}(\underline{\mathbf{x}}), \underline{T'}(\underline{\mathbf{x}})\rangle = T'(\mathbf{u})$. This shows that T^* coincides with the evaluation $\mathbf{e}_\mathbf{u}$.

Thus if $T = \sum_i \omega_i\, \mathbf{u}_{i,1}^{\delta_1}\mathbf{u}_{i,2}^{\delta_2}\dots\mathbf{u}_{i,k}^{\delta_k}$, then T^* coincides with the weighted sum of evaluations $T^* = \sum_i \omega_i\, \mathbf{e}_{bu_i}$ on $R_{\delta_1,\delta_2,\dots,\delta_k}$. We reduce the decomposition problem of T to the decomposition of T^* as a weighted sum of evaluations $T^* = \sum_i \omega_i\, \mathbf{e}_{bu_i}$.

2.1 Solving Polynomial Equations by Eigenvector Computation

We recall the definition of quotient algebra A of the ring of polynomials in a collection of variables $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k]$ where $\mathbf{x}_j = [\mathbf{x}_{j,1}, \mathbf{x}_{j,2}, \dots, \mathbf{x}_{j,n_j}]$ for $j = 1, \dots, k$ by an ideal I of multivariate polynomials. We use eigen-structure properties of multiplication operators and their transpose in a chosen basis of A and its dual to compute the x's coordinates of points in the decomposition of the associated multi-linear tensor.

A quotient algebra $\mathcal{A} = \mathbb{C}[\mathbf{x}]/I$ is *Artinian* if it is of finite dimension over $\mathbb{C}$. In the case of partial symmetric tensor, the variables $\mathbf{x}$ and $\mathbf{y}$ are divided into bunches of sub-variables such that $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k]$ and $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_k]$. Hereafter, in the case of multi-symmetric tensor the ideal I defines a finite number of roots $\mathcal{V}(I) = \{\xi_1, \xi_2, \dots, \xi_{r'}\} = \{\xi \in \mathbb{C}^n \mid \forall q \in I, q(\xi) = 0\}$ where $n = \sum_{j=1}^k (n_j + 1)$ such that $n_j + 1$ is the dimension of each vector space spanned by $\mathbf{x}_j$ and we have a decomposition of $\mathcal{A}$ as a sum of sub-algebras:

$$\mathcal{A} = \mathbb{C}[\mathbf{x}]/I = \mathcal{A}_1 \oplus \dots \oplus \mathcal{A}_{r'}$$

where $\mathcal{A}_p = \mathbf{u}_{\xi_p}\mathcal{A} \sim \mathbb{C}[\mathbf{x}]/Q_p$ and Q_p is the primary component of I associated to the root $\xi_p \in \mathbb{C}^n$. The elements $\mathbf{u}_1, \dots, \mathbf{u}_{r'}$ satisfy the relations

$$\mathbf{u}_{\xi_p}^2(\mathbf{x}) \equiv \mathbf{u}_{\xi_p}(\mathbf{x}),\ \sum_{i=1}^{r} \mathbf{u}_{\xi_p}(\mathbf{x}) \equiv 1.$$

The polynomials $\mathbf{u}_{\xi_1}, \dots, \mathbf{u}_{\xi_{r'}}$ are called *idempotents* of $\mathcal{A}$. The dimension of $\mathcal{A}_p$ is the *multiplicity* of the point ξ_p. For more details, see [6][Chap. 4].

For $g \in \mathbb{C}[\mathbf{x}]$, the multiplication operator $\mathcal{M}_g$ is defined by

$$\begin{aligned} \mathcal{M}_g : \mathcal{A} &\to \mathcal{A} \\ h &\mapsto \mathcal{M}_g(h) = g\,h. \end{aligned}$$

The transpose $\mathcal{M}_g^\intercal$ of the multiplication operator $\mathcal{M}_g$ is

$$\begin{aligned} \mathcal{M}_g^\intercal : \mathcal{A}^* &\to \mathcal{A}^* \\ \Lambda &\mapsto \mathcal{M}_g^\intercal(\Lambda) = \Lambda \circ \mathcal{M}_g = g \star \Lambda. \end{aligned}$$

The main property that we will use to recover the roots is the following [6][Theorem 4.23].

Proposition 2. *Let I be an ideal of $\mathbb{C}[\mathbf{x}]$ and suppose that $\mathcal{V}(I) = \{\xi_1, \xi_2, \ldots, \xi_{r'}\}$. Then*

- *for all $g \in \mathcal{A}$, the eigenvalues of $\mathcal{M}_g$ and $\mathcal{M}_g^\intercal$ are the values $g(\xi_1), \ldots, g(\xi_{r'})$ of the polynomial g at the roots with multiplicities $\mu_p = \dim \mathcal{A}_p$.*
- *The eigenvectors common to all $\mathcal{M}_g^\intercal$ with $g \in \mathcal{A}$ are - up to a scalar - the evaluations $\mathbf{e}_{\xi_1}, \ldots, \mathbf{e}_{\xi_{r'}}$.*

If $B = \{b_1, \ldots, b_r\}$ is a basis of $\mathcal{A}$, then the coefficient vector of the evaluation $\mathbf{e}_{\xi_p}$ in the dual basis of B is $\left[\langle \mathbf{e}_{\xi_p} | b_j \rangle\right]_{\beta \in B} = [b_j(\xi_p)]_{p=1\ldots r} = B(\xi_p)$. The previous proposition says that if M_g is the matrix of $\mathcal{M}_g$ in the basis B of $\mathcal{A}$, then

$$M_g^\intercal \, B(\xi_p) = g(\xi_p) \, B(\xi_p).$$

If moreover the basis B contains the monomials $1, x_{1,1}, x_{1,2}, \ldots, x_{1,n_1}$, then the common eigenvectors of $M_g^\intercal$ are of the form $\mathbf{v}_p = c\,[1, \xi_{p,1,1}, \ldots, \xi_{p,1,n_1}, \ldots]$ and the x's coordinates of the root ξ_p can be computed from the coefficients of $\mathbf{v}_p$ by taking the ratio of the coefficients of the monomials $x_{1,1}, \ldots, x_{1,n_1}$ by the coefficient of 1: $\xi_{p,1,i_1} = \frac{\mathbf{v}_{p,1,i_1+1}}{\mathbf{v}_{p,1,1}}$. Thus computing the common eigenvectors of all the matrices $M_g^\intercal$ for $g \in \mathcal{A}$ yield the x's coordinates of the roots ξ_p $(p = 1, \ldots, r)$.

In practice, it is enough to compute the common eigenvectors of $M_{x_{1,1}}^\intercal, \ldots, M_{x_{1,n_1}}^\intercal$, since $\forall g \in \mathbb{C}[\mathbf{x}_1], M_g^\intercal = g(M_{x_{1,1}}^\intercal, \ldots, M_{x_{1,n_1}}^\intercal)$. Therefore, the common eigenvectors $M_{x_{1,1}}^\intercal, \ldots, M_{x_{1,n_1}}^\intercal$ are also eigenvectors of any $M_g^\intercal$.

The multiplicity structure, that is the dual $Q_p^\perp$ of each primary component Q_p of I, also called the *inverse system* of the point ξ_p can be deduced by linear algebra tools (see e.g. [13]).

In the case of simple roots, we have the following property [6][Chap. 4]:

Proposition 3. *If the roots $\{\xi_1, \xi_2, \ldots, \xi_r\}$ of I are simple (i.e. $\mu_p = \dim \mathcal{A}_p = 1$) then we have the following:*

- *$\mathfrak{u} = \{\mathbf{u}_{\xi_1}, \ldots, \mathbf{u}_{\xi_r}\}$ is a basis of $\mathcal{A}$.*
- *The polynomials $\mathbf{u}_{\xi_1}, \ldots, \mathbf{u}_{\xi_r}$ are interpolation polynomials at the roots ξ_p: $\mathbf{u}_{\xi_p}(\xi_q) = 1$ if $p = q$ and 0 otherwise.*
- *The matrix of $\mathcal{M}_g$ in the basis $\mathfrak{u}$ is the diagonal matrix $\operatorname{diag}(g(\xi_1), \ldots, g(\xi_r))$.*

This proposition tells us that if g is separating the roots, i.e. $g(\xi_p) \neq g(\xi_q)$ for $p \neq q$, then the eigenvectors of $\mathcal{M}_g$ are, up to a scalar, interpolation polynomials at the roots.

2.2 Artinian Gorenstein Algebra of a Multivariate Hankel Operator

In this section, we detail the construction of the quotient algebra A_τ by the kernel I_τ of the Hankel operator H_τ associated to the dual of the tensor T. We compute a basis of A_τ such that the submatrix associated to it has a maximal non-zero minor of a truncated matrix of H_τ. We recall how to compute the multiplication matrices in this associated basis and its dual using some shifted

submatrices of H_τ. We notice that not all of them are easy to compute. We benefit from properties of generalized eigenvalues of multiplication matrices by $y's$ to compute the x's coordinates of points. We show how to use the generalized eigenvectors of the multiplication matrices to compute the weights.

We associate to a Hankel operator H_τ, the quotient $\mathcal{A}_\tau = \mathbb{C}[\mathbf{x}]/I_\tau$ of the polynomial ring $\mathbb{C}[\mathbf{x}]$ modulo the kernel $I_\tau = \{p \in \mathbb{C}[\mathbf{x}] \mid \forall q \in R, \langle \tau \mid pq \rangle = 0\}$ of H_τ. We check that I_τ is an ideal of $\mathbb{C}[\mathbf{x}]$, so that $\mathcal{A}_\tau$ is an algebra.

As $\mathcal{A}_\tau = \mathbb{C}[\mathbf{x}]/I_\tau \sim \operatorname{img} H_\tau$, the operator H_τ is of finite rank r, if and only if, $\mathcal{A}_\tau$ is Artinian of dimension $\dim_{\mathbb{C}} \mathcal{A}_\tau = r$.

A quotient algebra $\mathcal{A}$ is called *Gorenstein* if its dual $\mathcal{A}^* = \operatorname{Hom}_{\mathbb{C}}(\mathcal{A}, \mathbb{C})$ is a free $\mathcal{A}$-module generated by one element.

In our context, we have the following equivalent properties [14]:

- $\tau = \sum_{p=1}^{r'} \omega_p(\mathbf{y}) \mathbf{e}_{\xi_p}(\mathbf{y})$ with $\omega_p \in \mathbb{C}[\mathbf{y}]$, $\xi_p \in \mathbb{C}^n$ and $\sum_{p=1}^{r'} \mu(\omega_p) = r$ where $n = \sum_{j=1}^{k}(n_j + 1)$,
- H_τ is of rank r,
- $\mathcal{A}_\tau$ is an Artinian Gorenstein algebra of dimension r.

Another property that will be helpful to determine a basis of $\mathcal{A}_\tau$ is the following:

Lemma 2. *Let $B = \{b_1, \ldots, b_r\}$, $B' = \{b'_1, \ldots, b'_r\} \subset \mathbb{C}[\mathbf{x}]$. If the matrix $H_\tau^{B,B'} = (\langle \tau | b_p b'_q \rangle)_{1 \leq p,q \leq r}$ is invertible, then B and B' are linearly independent in $\mathcal{A}_\tau$.*

By this Lemma, bases of $\mathcal{A}_\tau$ can be computed by identifying non-zero minors of maximal size of the matrix of H_τ.

Proposition 4. *Let B, B' be basis of $\mathcal{A}_\tau$ and $g \in \mathbb{C}[\mathbf{x}]$. We have*

$$H_{g\star\tau}^{B,B'} = (M_g^B)^\intercal H_\tau^{B,B'} = H_\tau^{B,B'} M_g^{B'}. \tag{3}$$

where M_g^B (resp. $M_g^{B'}$) is the matrix of the multiplication by g in the basis B (resp. B') of $\mathcal{A}_\tau$.

We deduce the following property:

Proposition 5. *Let $\tau(\mathbf{y}) = \sum_{p=1}^{r} \omega_p(\mathbf{y}) \mathbf{e}_{\xi_p}(\mathbf{y})$ with $\omega_p \in \mathbb{C}[\mathbf{y}] \backslash \{0\}$ and $\xi_p \in \mathbb{C}^n$ distinct and let B, B' be bases of $\mathcal{A}_\tau$. We have the following properties:*

- *For $g \in \mathbb{C}[\mathbf{x}]$, $M_g^{B'} = (H_\tau^{B,B'})^{-1} H_{g\star\tau}^{B,B'}$, $(M_g^B)^\intercal = H_{g\star\tau}^{B,B'} (H_\tau^{B,B'})^{-1}$.*
- *For $g \in \mathbb{C}[\mathbf{x}]$, the generalized eigenvalues of $(H_{g\star\tau}^{B,B'}, H_\tau^{B,B'})$ are $g(\xi_p)$ with multiplicity $\mu_p = \mu(\omega_p)$, $p = 1, \ldots, r$.*
- *The generalized eigenvectors common to all $(H_{g\star\tau}^{B,B'}, H_\tau^{B,B'})$ for $g \in \mathbb{C}[\mathbf{x}]$ are - up to a scalar - $(H_\tau^{B,B'})^{-1} B(\xi_p)$, $p = 1, \ldots, r$.*

Proof. The two first points are direct consequences of Propositions 4 and 2. The third point is also a consequence of Proposition 2, since the coordinate vector of the evaluation $\mathbf{e}_{\xi_p}$ in the dual basis of B is $B(\xi_p)$ for $p = 1, \ldots, r$.

This proposition shows that the matrices of multiplication by an element g in $\mathcal{A}$, and thus the roots $\{\xi_1, \ldots, \xi_r\}$ and their multiplicity structure, can be computed from truncated Hankel matrices, provided we can determine bases B, B' of $\mathcal{A}_\tau$. In practice, it is enough to compute the generalized eigenvectors common to $(H^{B,B'}_{x_{1,i_1}\star\tau}, H^{B,B'}_{\tau})$ for $i_1 = 1, \ldots, n_1$ to recover the roots. As $H^{B,B'}_{x_{1,i_1}\star\tau} = H^{x_{1,i_1}B,B'}_{\tau} = H^{B,x_{1,i_1}B'}_{\tau}$, the decomposition can be computed from sub-matrices of H^{B,B'^+}_{τ} or $H^{B^+,B'}_{\tau}$ where $B^+ = B \cup x_{1,1}B \cup \cdots \cup x_{1,n_1}B$, $B'^+ = B' \cup x_{1,1}B' \cup \cdots \cup x_{1,n_1}B'$.

3 Multilinear Tensor Decomposition Problem

In this section, we analyze the easiest case of multi symmetric tensor where it is of degree one at each bunch of sub-variables. Our goal is to decompose τ which is equal to T^* as a weighed sum of evaluations by computing the eigenstructure of $\mathcal{A}_\tau$ which is based on the computation of multiplication operators. We simplify notations by using subscripts of variables and coefficients instead of multi-index exponents. We compute the truncated Singular Value Decomposition of a generic linear combination of a shifted Hankel matrices by the first collection of variables. By linearity and properties of the multiplication operators by one variable described in Sect. 2, we deduce the multiplication operators by more complex variables which could be used to compute weights and points.

We choose two monomial bases B_1 and B_2 indexing respectively rows and columns of the Hankel matrix $H^{B_1,B_2}_{T^*}$ associated to the tensor T^*, such that the set of monomials $\{B_1 * B_2 x_{j,i_j}, 0 \leq i_j \leq n_j, 1 \leq j \leq k\}$ span the set of deshomogenized polynomials $R_{\delta_1,\delta_2,\ldots,\delta_k}$.

The matrix of the truncated Hankel operator in the basis B_1 and the dual basis B_2 is $H^{B_1,B_2}_{T^*} = [t_{i_1,i_2,\ldots,i_k}]_{\substack{0\leq i_1\leq n_1 \\ 0\leq i_2\leq n_2 \\ \vdots \\ 0\leq i_k\leq n_k}}$.

The Hankel matrix associated to the tensor $x_{1,i_1} * T^*$ is defined as $H_{1,i_1} = H^{B_1,B_2}_{x_{1,i_1}*T^*} = H^{x_{1,i_1}*B_1,B_2}_{T^*} = [t_{\alpha+\beta}]_{\alpha\in x_{1,i_1}*B_1,\beta\in B_2}$, all the elements of the matrix are divisible in x_{1,i_1} and of degree δ.

For example, the Hankel matrix associated to $x_1 * T^*$ in the monomials basis B_1 and B_2 is denoted by H_0. Let $\lambda(\mathbf{x}_1) = \lambda_0 + \lambda_1 x_{1,1} + \ldots + \lambda_{n_1} x_{1,n_1}$ is a linear form with generic chosen coefficients $\lambda_{i_1}, i_1 = 0, \ldots, n_1$, we build a linear combination of $H_{1,i_1}, i_1 = 0, \ldots, n_1$ such that $\widehat{H}_0 = \sum_{i_1=0}^{n_1} \lambda_{i_1} H_{1,i_1}$ we compute the singular value decomposition of it.

Computing the singular value decomposition of $\widehat{H}_0$, we obtain

$$\widehat{H}_0 = USV^\intercal$$

where S is the diagonal matrix of all singular values of $\widehat{H}_0$ arranged in a decreasing order, U is an unitary matrix whose columns are the left singular vectors of $\widehat{H}_0$, V is an unitary matrix whose columns are the right singular vectors of $\widehat{H}_0$.

We denote by U^{H} the hermitian transpose of U and $\overline{V}$ the conjugate of V. We denote by U_r and V_r the truncated matrices of the first r columns of U and V and S_r the diagonal matrix of the first r rows and r columns of S.

We denote $\overline{B}_1 = \langle 1, x_{1,1}, \ldots, x_{1,n_1} \rangle$ and $\overline{B}_2 = \langle 1, x_{k,1}, \ldots, x_{k,n_k} \rangle$. Let $u_i = [u_{\alpha,i}]_{\alpha \in \overline{B}_1}$ and $v_j = [v_{\beta,j}]_{\beta \in \overline{B}_2}$ be respectively the i^{th} and j^{th} columns of U^{H} and $\overline{V}$. We denote by $u_i(\mathbf{x}_1) = u_i^{\mathsf{T}} U_r^{\mathsf{H}}$ and $v_j(\mathbf{x}_1) = v_j^{\mathsf{T}} \overline{V}_r$ the corresponding polynomials. The bases formed by these first r polynomials are denoted $U_r^{\mathsf{H}} := (u_i(\mathbf{x}_1))_{i=1,\ldots,r}$ and $\overline{V}_r := (v_j(\mathbf{x}_1))_{j=1,\ldots,r}$. We will also denote by U_r^{H} (resp. $\overline{V}_r$) the corresponding coefficient matrix, formed by the first rows (resp. columns) of U^{H} (resp. $\overline{V}$). We denote by S_r the diagonal matrix of the first r rows and columns of S, formed by the first r singular values.

We denote by H_0^r, H_{1,i_1}^r and $\widehat{H}_0^r$ the matrices obtained by the truncated singular value decomposition of H_0 ,H_{i_1} and $\widehat{H}_0$ respectively.

We have the following property

$$H_{i_1}^r = (M_{x_{1,i_1}}^{U_r^{\mathsf{H}}})^{\mathsf{T}} H_0^r = H_0^r M_{x_{1,i_1} * T}^{\overline{V}_r}$$

where $M_{x_{1,i_1}}^{U_r^{\mathsf{H}}}$ (resp. $M_{x_{1,i_1}}^{\overline{V}_r}$) is the multiplication matrix by x_{1,i_1} in the basis U_r^{H} (resp. $\overline{V}_r$) and $M_{x_{1,i_1} * T}^{\overline{V}_r}$ is the multiplication matrix by $x_{1,i_1} * T$ in the basis $\overline{V}_r$. Then by linearity, we obtain $\widehat{H}_0^r = \sum_{i_1=0}^{n_1} \lambda_{i_1} H_{1,i_1}^r = H_0^r \sum_{i_1=0}^{n_1} \lambda_{i_1} M_{x_{1,i_1} * T}^{\overline{V}_r} = H_0^r M_{\lambda(\mathbf{x}_1) * T}^{\overline{V}_r}$.

Then $(\widehat{H}_0^r)^{-1} = (M_{\lambda(\mathbf{x}_1) * T}^{\overline{V}_r})^{-1} (H_0^r)^{-1}$ so multiplying by the first equation we get

$$(\widehat{H}_0^r)^{-1} H_{1,i_1}^r = (M_{\lambda(\mathbf{x}_1) * T}^{\overline{V}_r})^{-1} M_{x_{1,i_1} * T}^{\overline{V}_r} = M_{(x_{1,i_1}/\lambda(\mathbf{x}_1)) * T}^{\overline{V}_r}$$

We compute the eigenvalues and the eigenvectors of the multiplication matrices $M_{(x_{1,i_1}/\lambda(\mathbf{x}_1)) * T}^{\overline{V}_r}$ in order to obtain the weights and the points of the decomposition.

3.1 Algorithm

We describe now the algorithm to recover the sum $\underline{T}^*(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}}) = \sum_{p=1}^{r} \omega_p \mathbf{e}_{\mathbf{u}_p}(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}})$, $\omega_p \in \mathbb{C} \setminus \{0\}, \mathbf{u}_p \in \mathbb{C}^{\sum_{l=1}^{3}(n_l+1)}$, from the moments of degree at most one at each bunch of coordinates $(t_{i,j,k})_{\substack{0 \le i \le n_1 \\ 0 \le j \le n_2 \\ 0 \le k \le n_3}}$ of the formal power series. To simplify, we change notations to better understand the nine dimensional multivariate space seen as three dimensional space. We only use 3 bunches of variables such that x_{1,i_1} by x_i and x_{2,i_2} by y_j and x_{3,i_3} by z_k.

Algorithm 3.1. Decomposition of polynomial-exponential series with constant weights

Input: the moments $(t_{i,j,k})_{\substack{0\le i\le n_1\\ 0\le j\le n_2\\ 0\le k\le n_3}}$ of T^*.

1. Compute the monomial sets $A_1 = (x_i y_j)_{\substack{0\le i\le n_1\\ 0\le j\le n_2}}$ and $A_2 = (z_0, z_1, \ldots, z_{n_3})$ and substitute the x_0, y_0 and z_0 by 1 to define B_1 and B_2.
2. Compute the Hankel matrix $H_{T^*}^{B_1,B_2} = [t_{i,j,k}]_{\substack{0\le i\le n_1\\ 0\le j\le n_2\\ 0\le k\le n_3}}$ for the monomial sets B_1 and B_2.
3. Compute the singular value decomposition of $H_{T^*}^{\overline{B}_1,\overline{B}_2} = USV^{\mathsf{T}}$ where $\overline{B}_1 = \langle 1, x_1, \ldots, x_{n_1}\rangle$ and $\overline{B}_2 = \langle 1, z_1, \ldots, z_{n_3}\rangle$ with singular values $s_1 \ge s_2 \ge \cdots \ge s_m \ge 0$.
4. Determine its numerical rank, that is, the largest integer r such that $\frac{s_r}{s_1} \ge \epsilon$.
5. Form the multiplication matrices by y_j in the basis $\overline{V}_r$, $M_{y_j}^{\overline{V}_r} = S_r^{-1} U_r^{\mathsf{H}} H_{y_j \star T^*}^{\overline{B}_1,\overline{B}_2} \overline{V}_r$ where $H_{y_j \star T^*}^{\overline{B}_1,\overline{B}_2}$ is the Hankel matrix associated to $y_j \star T^*$ for $j = 1, \ldots, n_2$.
6. Compute the eigenvectors $\mathbf{v}_p$ of $\sum_{j=1}^{n_2} l_j M_{y_j}^{\overline{V}_r}$ such that $|l_j| \le 1, j = 1, \ldots, n_2$ and for each $p = 1, \ldots, r$ do the following:
   - The $y's$ coordinates of the $\mathbf{u}_p$ are the eigenvalues of the multiplication matrices by y_j. Use the formula $M_{y_j}^{\overline{V}_r}\mathbf{v}_p = u_{p,2,j}\mathbf{v}_p$ for $p = 1, \ldots, r$ and $j = 1, \ldots, n_2$ and deduce the $u_{p,2,j}$.
   - Write the matrix $H_{T^*}^{\overline{B}_1,\overline{B}_2}$ in the basis of interpolation polynomials(i.e. the eigenvectors $\mathbf{v}_p$) and use the corresponding matrix $\mathcal{T}$ to compute the $z's$ coordinates. Divide the $k^{\mathbf{th}}$ row on the first row of the matrix $\mathcal{T}$ to obtain the values of $u_{p,3,k}$ for $p = 1, \ldots, r$ and $k = 1, \ldots, n_3$.
   - The $x's$ coordinates of $\mathbf{u}_p$ are computed using the eigenvectors of the transpose of the matrix $M_{y_j}^{\overline{V}_r}$. They -are up to scalar- the evaluations, they are represented by vectors of the form $\mathbf{v}_p^* = \mu_p[1, u_{p,1,1}, \ldots, u_{p,1,n_1}]$. Compute $\mathbf{v}_p^*$ as the p^{th} column of the transpose of the inverse of the matrix $V = [v_1, \ldots, v_r]$ for $p = 1, \ldots, r$ and deduce $u_{p,1,i} = \frac{\mathbf{v}_p^*[i+1]}{\mathbf{v}_p^*[1]}$ for $p = 1, \ldots, r$ and $i = 1, \ldots, n_1$.
   - Compute $\omega_p = \frac{\langle T^* | \mathbf{v}_p\rangle}{\mathbf{v}_p(\mathbf{u}_p)}$.

Output: $r \in \mathbb{N}$, $\omega_p \in \mathbb{C}\backslash(0)$, $\mathbf{u}_p \in \mathbb{C}^{\sum_{l=1}^{3}(n_l+1)}$, $p = 1, \ldots, r$ such that $\underline{T}^*(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}}) = \sum_{p=1}^{r} \omega_p \mathbf{e}_{\mathbf{u}_p}(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}})$ up to degree one at each bunch of coordinates.

The cost of the SVD computation is in $\mathcal{O}(s^3)$ where $s \ge r$ is the maximal size of the Hankel matrix H_0 and r the rank of the decomposition. The computation of each multiplication matrice is in $\mathcal{O}(r^2)$ and the eigencomputation is in $\mathcal{O}(r^3)$. This yields a complexity bound in $\mathcal{O}(s^3 + n\,r^2)$ for the complete algorithm, where $n = \max(n_1, n_2, n_3)$ is a bound on the dimension of the spaces. This complexity bound extends to the decomposition of general multi-symmetric tensors, provided $r = \operatorname{rank} H_0$.

4 Example

In this section, we illustrate the decomposition algorithm on a multi-linear tensor of degree one at each bunch of 3 variables and of rank 3.

If $\delta_l = 1$ for all $l = 1, \ldots, k$, $k > 1$ and $n_l = n$, let $k = 3$, $n_l = n = 2$, $r = 3$ and $\delta_l = 1$ then we have $\mathbf{x} = (x_0, x_1, x_2)$, $\mathbf{y} = (y_0, y_1, y_2)$ and $\mathbf{z} = (z_0, z_1, z_2)$. For

$$\bar{\alpha} \in \mathbb{N}^3, |\bar{\alpha}| = 1 \Rightarrow \bar{\alpha} = (1), (0,1), (0,1) \Rightarrow \mathbf{x}^{\bar{\alpha}} = x_i, i = 0, \ldots, 2$$
$$\bar{\beta} \in \mathbb{N}^3, |\bar{\beta}| = 1 \Rightarrow \bar{\beta} = (1), (0,1), (0,1) \Rightarrow \mathbf{y}^{\bar{\beta}} = y_j, j = 0, \ldots, 2$$
$$\bar{\gamma} \in \mathbb{N}^3, |\bar{\gamma}| = 1 \Rightarrow \bar{\gamma} = (1), (0,1), (0,1) \Rightarrow \mathbf{z}^{\bar{\gamma}} = z_k, k = 0, \ldots, 2$$

The multi symmetric tensor is defined by a multi symmetric array of coefficients such that $t_{\alpha,\beta,\gamma} := t_{\bar{\alpha},\bar{\beta},\bar{\gamma}} = t_{i,j,k}$ then $T(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{\substack{0 \le i \le 2 \\ 0 \le j \le 2 \\ 0 \le k \le 2}} t_{i,j,k} x_i y_j z_k =$

$$\begin{aligned}
&0.4461757334 x_0 y_0 z_0 - 0.2262004083 x_0 y_0 z_1 + 0.4427031740 x_0 y_0 z_2 \\
&- 0.2756785277 x_0 y_1 z_0 + 0.1612318550 x_0 y_1 z_1 - 0.3100164212 x_0 y_1 z_2 \\
&- 0.1209490221 x_0 y_2 z_0 + 0.1465160338 x_0 y_2 z_1 - 0.1169341103 x_0 y_2 z_2 \\
&- 0.01239930649 x_1 y_0 z_0 - 0.05189330981 x_1 y_0 z_1 + 0.01803564422 x_1 y_0 z_2 \\
&- 0.01336683543 x_1 y_1 z_0 + 0.02632784503 x_1 y_1 z_1 - 0.02598209626 x_1 y_1 z_2 \\
&- 0.3195263612 x_1 y_2 z_0 + 0.09311605022 x_1 y_2 z_1 - 0.1116610246 x_1 y_2 z_2 \\
&- 0.1460187051 x_2 y_0 z_0 + 0.06557223848 x_2 y_0 z_1 - 0.1734312692 x_2 y_0 z_2 \\
&+ 0.1010145926 x_2 y_1 z_0 - 0.05743078561 x_2 y_1 z_1 + 0.1238292801 x_2 y_1 z_2 \\
&- 0.1485221955 x_2 y_2 z_0 + 0.03231415762 x_2 y_2 z_1 - 0.376925099 e - 2 x_2 y_2 z_2.
\end{aligned}$$

Let $x_0 = y_0 = z_0 = 1$ then $\underline{T}(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}}) = \sum_{\substack{1 \le i \le 2 \\ 1 \le j \le 2 \\ 1 \le k \le 2}} t_{i,j,k} \underline{x}_i \underline{y}_j \underline{z}_k$

Then the tensor decomposition problem is $\underline{T}(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}}) = \sum_{p=1}^{r} \omega_p \mathbf{u}_{p,1}(\underline{\mathbf{x}}) \mathbf{u}_{p,2}(\underline{\mathbf{y}}) \mathbf{u}_{p,3}(\underline{\mathbf{z}})$. Given all the moments of degree at most one at each bunch of coordinates $(t_{i,j,k})_{\substack{0 \le i \le 2, \\ 0 \le j \le 2 \\ 0 \le k \le 2}}$,

We create two sets $A_1 = (\mathbf{x}^{\bar{\alpha}} \mathbf{y}^{\bar{\beta}})_{\substack{|\bar{\alpha}|=1 \\ |\bar{\beta}|=1}} = (x_i y_j)_{\substack{0 \le i \le 2 \\ 0 \le j \le 2}}$ and $A_2 = (\mathbf{z}^{\bar{\gamma}})_{|\bar{\gamma}|=1} = (z_k)_{0 \le k \le 2}$ so that

$A_1 = (x_0 y_0, x_0 y_1, x_0 y_2, x_1 y_0, x_1 y_1, x_1 y_2, x_2 y_0, x_2 y_1, x_2 y_2)$ and $A_2 = (z_0, z_1, z_2)$.

For $x_0 = y_0 = z_0 = 1$ then $B_1 = (1, y_1, y_2, x_1, x_1 y_1, x_2, x_2 y_1, x_2 y_2)$ and $B_2 = (1, z_1, z_2)$, the Hankel matrix associated to the tensor in the monomial basis B_1 and B_2 is

$$H_{T^*}^{B_1,B_2} = [t_{\bar{\alpha}+\bar{\beta}+\bar{\gamma}}]_{\substack{|\bar{\alpha}|=1\\|\bar{\beta}|=1\\|\bar{\gamma}|=1}} = \begin{array}{c} \begin{matrix} 1 & y_1 & y_2 & x_1 & x_1y_1 & x_1y_2 & x_2 & x_2y_1 & x_2y_2 \end{matrix} \\ \begin{bmatrix} t_0 & t_1 & t_2 & t_{0,1} & t_{0,2} & t_{1,1} & t_{1,2} & t_{2,1} & t_{2,2} \\ t_{0,1} & t_{1,1} & t_{2,1} & t_{0,1,1} & t_{0,2,1} & t_{1,1,1} & t_{1,2,1} & t_{2,1,1} & t_{2,2,1} \\ t_{0,2} & t_{1,2} & t_{2,2} & t_{1,2} & t_{0,2,2} & t_{1,1,2} & t_{1,2,2} & t_{2,1,2} & t_{2,2,2} \end{bmatrix} \end{array} \begin{matrix} 1 \\ z_1 \\ z_2 \end{matrix}$$

$$= \begin{array}{c} \begin{matrix} 1 & y_1 & y_2 & x_1 & x_1y_1 & x_1y_2 & x_2 & x_2y_1 & x_2y_2 \end{matrix} \\ \begin{bmatrix} 0.4461757 & 0.2756785 & 0.1209490 & 0.01239930 & 0.01336683 & 0.3195263 & 0.14601870 & 0.1010145 & 0.1485221 \\ -0.2262004 & 0.1612318 & 0.1465160 & 0.05189330 & 0.02632784 & 0.09311605 & 0.06557223 & 0.05743078 & 0.03231415 \\ 0.4427031 & 0.3100164 & 0.1169341 & .01803564 & 0.02598209 & 0.1116610 & 0.1734312 & 0.1238292 & 0.00376925 \end{bmatrix} \end{array}$$

All the entries of this matrix are known, we choose $\overline{B}_1 = \langle 1, x_1, x_2 \rangle$ and $\overline{B}_2 = \langle 1, z_1, z_2 \rangle$ to be able to multiply by y_1 and to compute the multiplication matrix. Computing the singular value decomposition of $H_{T^*}^{\overline{B}_1,\overline{B}_2}$, we obtain

$$H_{T^*}^{\overline{B}_1,\overline{B}_2} = USV^{\intercal} = \begin{array}{c} \begin{matrix} 1 & x_1 & x_2 \end{matrix} \\ \begin{bmatrix} t_0 & t_1 & t_2 \\ t_{0,1} & t_{1,1} & t_{2,1} \\ t_{0,2} & t_{1,2} & t_{2,2} \end{bmatrix} \end{array} \begin{matrix} 1 \\ z_1 \\ z_2 \end{matrix} = \begin{array}{c} \begin{matrix} 1 & x_1 & x_2 \end{matrix} \\ \begin{bmatrix} 0.4461757 & -0.01239930 & -0.1460187 \\ -0.2262004 & -0.05189330 & 0.06557223 \\ 0.4427031 & 0.01803564 & -0.1734312 \end{bmatrix} \end{array}$$

where S is the diagonal matrix of all singular values of $H_{T^*}^{\overline{B}_1,\overline{B}_2}$ arranged in a decreasing order, U is an unitary matrix whose columns are the left singular vectors of $H_{T^*}^{\overline{B}_1,\overline{B}_2}$, V is an unitary matrix whose columns are the right singular vectors of $H_{T^*}^{\overline{B}_1,\overline{B}_2}$. We denote by U^{H} the Hermitian transpose of U and $\overline{V}$ the conjugate of V.

Let $v_i = [v_{\alpha,i}]_{\alpha\in\overline{B}_1}$ and $w_j = [w_{\beta,j}]_{\beta\in\overline{B}_2}$ be respectively the i^{th} and j^{th} columns of U^{H} and $\overline{V}$. We denote by $v_i(\mathbf{x}) = v_i^{\intercal} U_r^{\mathsf{H}}$ and $w_j(\mathbf{z}) = w_j^{\intercal}\overline{V}_r$ the corresponding polynomials. The bases formed by these first r polynomials are denoted $U_r^{\mathsf{H}} := (v_i(\mathbf{x}))_{i=1,\ldots,r}$ and $\overline{V}_r := (w_j(\mathbf{z}))_{j=1,\ldots,r}$. We will also denote by U_r^{H} (resp. $\overline{V}_r$) the corresponding coefficient matrix, formed by the first rows (resp. columns) of U^{H} (resp. $\overline{V}$). We denote by S_r the diagonal matrix of the first r rows and columns of S, formed by the first r singular values. To compute the multiplication matrices $M_{y_1}^{\overline{V}_r}$ and $M_{y_2}^{\overline{V}_r}$ we need to compute the following matrices

$$H_{y_1*T^*}^{\overline{B}_1,\overline{B}_2} = \begin{array}{c} \begin{matrix} y_1 & y_1x_1 & x_2y_1 \end{matrix} \\ \begin{bmatrix} t_{0,1} & t_{1,1} & t_{2,1} \\ t_{0,1,1} & t_{1,1,1} & t_{2,1,1} \\ t_{0,1,2} & t_{1,1,2} & t_{2,1,2} \end{bmatrix} \end{array} \begin{matrix} 1 \\ z_1 \\ z_2 \end{matrix} = \begin{array}{c} \begin{matrix} y_1 & y_1x_1 & x_2y_1 \end{matrix} \\ \begin{bmatrix} -0.2756785 & -0.01336683 & 0.1010145 \\ 0.1612318 & 0.02632784 & -0.05743078 \\ -0.3100164 & -0.02598209 & 0.1238292 \end{bmatrix} \end{array}$$

$$H_{y_2*T^*}^{\overline{B}_1,\overline{B}_2} = \begin{array}{c} \begin{matrix} y_2 & y_2x_1 & x_2y_2 \end{matrix} \\ \begin{bmatrix} t_{0,2} & t_{1,2} & t_{2,2} \\ t_{0,2,1} & t_{1,2,1} & t_{2,2,1} \\ t_{0,2,2} & t_{1,2,2} & t_{2,2,2} \end{bmatrix} \end{array} \begin{matrix} 1 \\ z_1 \\ z_2 \end{matrix} = \begin{array}{c} \begin{matrix} y_2 & y_2x_1 & x_2y_2 \end{matrix} \\ \begin{bmatrix} -0.1209490 & -0.3195263 & -0.1485221 \\ 0.1465160 & 0.09311605 & 0.03231415 \\ -0.1169341 & -0.1116610 & -0.00376925 \end{bmatrix} \end{array}$$

Then we compute $M_{y_1}^{\overline{V}_r} = S_r^{-1} U_r^{\mathrm{H}} H_{y_1 * T^*}^{\overline{B}_1, \overline{B}_2} \overline{V}_r$ and $M_{y_2}^{\overline{V}_r} = S_r^{-1} U_r^{\mathrm{H}} H_{y_2 * T^*}^{\overline{B}_1, \overline{B}_2} \overline{V}_r$, and the eigenvectors $\mathbf{v}_p$ of $\sum_{j=1}^{2} l_j M_{y_j}^{\overline{V}_r}$ such that $|l_j| \leq 1, j = 1, \ldots, 2$. To recover the points $\mathbf{u}_p \in \mathbb{C}^{n*k}$ for $p = 1, \ldots, r$ of the form

$$\mathbf{u}_p = \begin{bmatrix} u_{p,1,1} & x_1 \\ u_{p,1,2} & x_2 \\ \\ u_{p,2,1} & y_1 \\ u_{p,2,2} & y_2 \\ \\ u_{p,3,1} & z_1 \\ u_{p,3,2} & z_2 \end{bmatrix}$$

We do the following:

In general we have $M_{x_{j,i_j}}^{\overline{V}_r} \mathbf{v}_i = u_{i,j,i_j} \mathbf{v}_i$, for $i = 1, \ldots, r$, $j = 1, \ldots, k$, $i_j = 1, n_j$ so in this case we get $M_{y_1}^{\overline{V}_r} \mathbf{v}_p = u_{p,2,1} \mathbf{v}_p$ and $M_{y_2}^{\overline{V}_r} \mathbf{v}_p = u_{p,2,2} \mathbf{v}_p$ for $p = 1, \ldots, 3$ so we compute $u_{p,2,1}$ and $u_{p,2,2}$ for $p = 1, \ldots, 3$. So that we get

$$\begin{aligned} u_{p,2,1} &= \begin{bmatrix} -0.746329870878 & -0.293761776025 & -0.304898408788 \end{bmatrix} \\ u_{p,2,2} &= \begin{bmatrix} 1.40328849510 & -0.336304368405 & -3.59031087599 \end{bmatrix}. \end{aligned}$$

The eigenvectors $\mathbf{v}_p \in \langle 1, x_1, x_2 \rangle$ for $p = 1, \ldots, 3$ are up to a scalar the interpolation polynomials at the roots so that if the dual of the tensor has an affine decomposition $\underline{T}^*(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}}) = \sum_{p=1}^{r} \omega_p \mathbf{e}_{\mathbf{u}_p}(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}})$ then $\underline{T}^*(\mathbf{v}_p) = \sum_{p=1}^{r} \omega_p \mathbf{e}_{\mathbf{u}_p}(\mathbf{v}_p) = \lambda_p \omega_p$, $\underline{T}^*(z_1 \mathbf{v}_p) = \sum_{p=1}^{r} \omega_p \mathbf{e}_{\mathbf{u}_p}(z_1 \mathbf{v}_p) = \lambda_p \omega_p u_{p,3,1}$ and $\underline{T}^*(z_2 \mathbf{v}_p) = \sum_{p=1}^{r} \omega_p \mathbf{e}_{\mathbf{u}_p}(z_2 \mathbf{v}_p) = \lambda_p \omega_p u_{p,3,2}$, for $p = 1, \ldots, 3$. Then the values of $u_{p,3,1}$ and $u_{p,3,2}$ for $p = 1, \ldots, 3$ come from the computation of the matrix:

$$\mathcal{T} = \begin{matrix} \\ \\ \\ \end{matrix} \begin{matrix} v_1 \\ \\ \\ \end{matrix} \begin{matrix} v_2 \\ \\ \\ \end{matrix} \begin{matrix} v_3 \\ \\ \\ \end{matrix} \begin{bmatrix} T^*(v_1) & T^*(v_2) & T^*(v_3) \\ T^*(z_1 v_1) & T^*(z_1 v_2) & T^*(z_1 v_3) \\ T^*(z_2 v_1) & T^*(z_2 v_2) & T^*(z_2 v_3) \end{bmatrix} \begin{matrix} 1 \\ z_1 \\ z_2 \end{matrix} = \begin{matrix} \\ \\ \\ \end{matrix} \begin{matrix} v_1 \\ \\ \\ \end{matrix} \begin{matrix} v_2 \\ \\ \\ \end{matrix} \begin{matrix} v_3 \\ \\ \\ \end{matrix} \begin{bmatrix} \lambda_1 \omega_1 & \lambda_2 \omega_2 & \lambda_3 \omega_3 \\ \lambda_1 \omega_1 u_{1,3,1} & \lambda_2 \omega_2 u_{2,3,1} & \lambda_3 \omega_3 u_{3,3,1} \\ \lambda_1 \omega_1 u_{1,3,2} & \lambda_2 \omega_2 u_{2,3,2} & \lambda_3 \omega_3 u_{3,3,2} \end{bmatrix} \begin{matrix} 1 \\ z_1 \\ z_2 \end{matrix}$$

Therefore the value of $u_{p,3,1}$ (resp. $u_{p,3,2}$) comes from the ratio of the second row (resp. the third row) and the first row of the matrix for $p = 1, \ldots, 3$. So that we get

$$\begin{aligned} u_{p,3,1} &= \begin{bmatrix} -0.655842579065 & 0.0321233423462 & -0.520955291 \end{bmatrix} \\ u_{p,3,2} &= \begin{bmatrix} 1.24749588143 & 0.403506877499 & 0.242728128570 \end{bmatrix}. \end{aligned}$$

The common eigenvectors of all $(M_{y_j}^{\overline{V}_r})^{\intercal}$ -are up to scalar- the evaluations, they are represented by vectors of the form $\mathbf{v}_p^* = \mu_p [1, u_{p,1,1}, u_{p,1,2}]$ in the dual basis of $\overline{B}_1 = \langle 1, x_1, x_2 \rangle$ then the computation of the coordinates of $u_{p,1,1}$ and

$u_{p,1,2}$ come from the eigenvectors of the transpose of multiplication operators which are obtained by transposing the inverse of the matrix V of vectors of $M_{y_j}^{\overline{V}^r}$ for $j = 1, \ldots, 2$, therefore the value of $u_{p,1,1}$ (resp. $u_{p,1,2}$) comes from the ratio of the second element of $\mathbf{v}_p^*$ (resp. the third element) and the first element of it, so that

$$u_{p,1,1} = \left[0.114279629148\ -1.08600705528\ 1.23814628617\right]$$
$$u_{p,1,2} = \left[-0.405714894278\ -0.567603220082\ 0.873482418287\right].$$

Notice that the computation of $\omega_p, p = 1, \ldots, 3$ can be done using the following formula $\omega_p = \frac{\langle T^* | \mathbf{v}_p \rangle}{\mathbf{v}_p(\mathbf{u}_p)}$ since if $\mathbf{v}_p \in \langle 1, x_1, x_2 \rangle$ then $\mathbf{v}_p = a_p + x_1 b_p + x_2 c_p$ and $\mathbf{v}_p^* = \mu_p[1, u_{p,1,1}, u_{p,1,2}] \in (\langle 1, x_1, x_2 \rangle)^*$, so that $\mathbf{v}_p(\mathbf{u}_p) = a_p + u_{p,1,1} b_p + u_{p,1,2} c_p = \langle \mathbf{v}_p | \mathbf{v}_p^* \rangle$, the computation gives

$$\omega = (\omega_p)_{1 \leq p \leq r} = \left[0.318579752246\ 0.08897389360312\ 0.0386220875736\right].$$

References

1. Anandkumar, A., Ge, R., Hsu, D., Kakade, S.M., Telgarsky, M.: Tensor decompositions for learning latent variable models (A Survey for ALT). In: Chaudhuri, K., Gentile, C., Zilles, S. (eds.) ALT 2015. LNCS, vol. 9355, pp. 19–38. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24486-0_2
2. Bernardi, A., Daleo, N.S., Hauenstein, J.D., Mourrain, B.: Tensor decomposition and homotopy continuation. Differential Geometry and its Applications, August 2017
3. Bernardi, A., Brachat, J., Comon, P., Mourrain, B.: General tensor decomposition, moment matrices and applications. J. Symbolic Comput. **52**, 51–71 (2013)
4. Brachat, J., Comon, P., Mourrain, B., Tsigaridas, E.P.: Symmetric tensor decomposition. Linear Algebra Appl. **433**(11–12), 1851–1872 (2010)
5. Comon, P., Jutten, C.: Handbook of Blind Source Separation: Independent Component Analysis and Applications. Academic press, Cambridge (2010)
6. Elkadi, M., Mourrain, B.: Introduction à la résolution des systèmes polynomiaux. Mathématiques et Applications, vol. 59. Springer, Heidelberg (2007)
7. Harmouch, J., Khalil, H., Mourrain, B.: Structured low rank decomposition of multivariate Hankel matrices. Linear Algebra Appl. (2017). https://doi.org/10.1016/j.laa.2017.04.015
8. Connelly, A., Tournier, J.D., Calamante, F.: Robust determination of the fiber orientation distribution in diffusion MRI: non-negativity constrained superresolved spherical deconvolution. NI **35**(4), 1459–1472 (2007)
9. Jiang, T., Sidiropoulos, N.D.: Kruskal's permutation lemma and the identification of candecomp/parafac and bilinear models with constant modulus constraints. IEEE Trans. Sig. Process. **52**(9), 2625–2636 (2004)
10. Landsberg, J.M.: Tensors: Geometry and Applications. Graduate Studies in Mathematics. American Mathematical Society, Providence (2011)
11. De Lathauwer, L., Castaing, J.: Tensor-based techniques for the blind separation of DS–CDMA signals. Sig. Process. **87**(2), 322–336 (2007). Tensor Signal Processing

12. Megherbi, T., Kachouane, M., Boumghar, F.O., Deriche, R.: Détection des croisements de fibre en IRM de diffusion par décomposition de tenseur: Approche analytique. In: Reconnaissance de Formes et Intelligence Artificielle (RFIA), France, June 2014
13. Mourrain, B.: Isolated points, duality and residues. J. Pure Appl. Algebra **117&118**, 469–493 (1996)
14. Mourrain, B.: Polynomial-exponential decomposition from moments. Found. Comput. Math. (2017). https://doi.org/10.1007/s10208-017-9372-x
15. Roch, S.: A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. IEEE/ACM Trans. Comput. Biol. Bioinform. **3**(1), Page 92 (2006)
16. Sanchez, E., Kowalski, B.R.: Tensorial resolution: a direct trilinear decomposition. J. Chemometr. **4**(1), 29–45 (1990)
17. Sidiropoulos, N.D., Giannakis, G.B., Bro, R.: Blind parafac receivers for DS-CDMA systems. IEEE Trans. Sig. Process. **48**(3), 810–823 (2000)
18. Smilde, A., Bro, R., Geladi, P.: Multi-way Analysis: Applications in the Chemical Sciences. Wiley, Chichester (2005)
19. Atkins, M.S., Weldeselassie, T.Y., Barmpoutis, A.: Symmetric positive semidefinite Cartesian tensor fiber orientation distributions (CT-FOD). Med. Image Anal. J. **16**(6), 1121–1129 (2012). Elsevier BV

Dimension Quasi-polynomials of Inversive Difference Field Extensions with Weighted Translations

Alexander Levin(✉)

The Catholic University of America, Washington, DC 20064, USA
levin@cua.edu
https://sites.google.com/a/cua.edu/levin

Abstract. We consider Hilbert-type functions associated with finitely generated inversive difference field extensions and systems of algebraic difference equations in the case when the translations are assigned positive integer weights. We prove that such functions are quasi-polynomials that can be represented as alternating sums of Ehrhart quasi-polynomials of rational conic polytopes. In particular, we generalize the author's results on difference dimension polynomials and their invariants to the case of inversive difference fields with weighted basic automorphisms.

Keywords: Inversive difference field · Inversive difference polynomials Characteristic set · Dimension quasi-polynomial

1 Introduction

The role of difference dimension polynomials in difference algebra is similar to the role of Hilbert polynomials in commutative algebra and algebraic geometry, as well as to the role of differential dimension polynomials in differential algebra. In particular, difference dimension polynomials and their invariants play the key role in the study of Krull-type dimension of difference and inversive difference modules and algebras, as well as of difference and inversive difference field extensions (see, for example, [8,10,11,13], and [9, Sects. 3.6, 4.6]). Furthermore, and this is probably the most important application of difference dimension polynomials outside of difference algebra, the difference dimension polynomial of a system of algebraic difference equations expresses the A. Einstein's strength of the system (see [9, Chap. 7] for a detailed description of this concept). In this connection, properties and methods of computation of difference dimension polynomials play a significant role in the qualitative theory of difference equations.

In this paper, we prove the existence and determine some invariants of a dimension quasi-polynomial associated with a finitely generated inversive difference field extension with weighted basic translations. We also show that such a quasi-polynomial is an alternating sum of Ehrhart quasi-polynomials associated with rational conic polytopes. Furthermore, we show that every "prime" system

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 67–80, 2017.
https://doi.org/10.1007/978-3-319-72453-9_5

of algebraic difference equations with weighted translations (that is, a system of the form $f_i(y_1, \ldots, y_n) = 0$, $i \in I$, where the left-hand sides generate a prime reflexive difference ideal in the corresponding ring of inversive difference polynomials) can be assigned a quasi-polynomial, which represents the Einstein's strength of the system. Note that systems of difference equations of these kind arise, in particular, from finite difference approximations of systems of PDEs with weighted derivatives, see, for example, [14,15]. One should also mention that our work continues the study of dimension quasi-polynomials associated with differential and difference algebraic structures initiated by C. Dönch in his dissertation [4]. In this work C. Dönch developed a Gröbner basis method for free difference-differential modules with weighted basic derivations and translations and used the developed technique to prove the existence of dimension quasi-polynomials associated with finitely generated modules over rings of difference-differential operators.

2 Preliminaries

In what follows $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{Z}_-$, $\mathbb{Q}$, and $\mathbb{R}$ denote the sets of all integers, non-negative integers, non-positive integers, rational numbers, and real numbers, respectively. If $m \in \mathbb{Z}$, $m \geq 1$, then by the product order on $\mathbb{N}^m$ we mean a partial order $<_P$ such that $(a_1, \ldots, a_m) <_P (a'_1, \ldots, a'_m)$ if and only if $a_i <_P a'_i$ for $i = 1, \ldots, m$.

By a ring we always mean an associative ring with unity. Every ring homomorphism is unitary (maps unity onto unity), every subring of a ring contains the unity of the ring.

A *difference ring* is a commutative ring R together with a finite set $\sigma = \{\alpha_1, \ldots, \alpha_m\}$ of mutually commuting endomorphisms of R. The set σ is called a *basic set* of R and the endomorphisms α_i are called *translations*. We also say that R is a σ-ring.

If $\alpha_1, \ldots, \alpha_m$ are automorphisms of R, we say that R is an *inversive difference ring* with the basic set σ. In this case the set $\{\alpha_1, \ldots, \alpha_m, \alpha_1^{-1}, \ldots, \alpha_m^{-1}\}$ is denoted by σ^* and R is also called a σ^*-ring. If a difference ring with a basic set σ is a field, it is called a difference (or σ-) field. If all α_i are automorphisms, it is called an inversive difference field or a σ^*-field. (We always use the upper index $*$ in the notation that refers to inversive algebraic difference structures; the corresponding notation without $*$ is common in the non-inversive case.)

In what follows we deal with inversive difference (σ^*-) rings and fields, where the basic set consists of m translations (automorphisms) $\alpha_1, \ldots, \alpha_m$. Furthermore, we will consider the free commutative group Γ generated by the set σ (we use the multiplicative notation, so that elements of Γ are power products of the form $\gamma = \alpha_1^{k_1} \ldots \alpha_m^{k_m}$ where $k_1, \ldots, k_m \in \mathbb{Z}$).

If R is a σ^*-ring and R_0 a subring of R such that $\alpha(R_0) \subseteq R_0$ for any $\alpha \in \sigma^*$, then R_0 is said to be an inversive difference (or σ^*-) subring of R; we also say that R is a σ^*-overring of R_0. In this case the restriction of α_i on R_0 ($1 \leq i \leq m$) is denoted by the same symbol α_i. If R is an inversive difference (σ^*-) field and R_0 a subfield of R which is also a σ^*-subring of R, then R_0 is said to be an inversive

difference (or σ^*-) subfield of R; R, in turn, is called an inversive difference (or σ^*-) field extension (or overfield) of R_0. In this case we also say that we have a σ^*-field extension R/R_0.

A subring (ideal) J of a σ^*-ring R is said to be a σ^*-subring of R (respectively, a σ^*-ideal of R) if J is closed with respect to the action of any mapping $\alpha_i \in \sigma^*$. If J is a σ^*-ideal of R, then the factor ring R/J has a natural structure of a σ^*-ring with the same basic set σ where $\alpha(a + I) = \alpha(a) + I$ for any coset $a + I \in R/I$ and $\alpha \in \sigma^*$. If a σ^*-ideal is prime, it is referred to as a *prime σ^*-ideal*.

If R is a σ^*-ring and $S \subseteq R$, then the intersection I of all σ^*-ideals of R containing the set S is the smallest σ^*-ideal of R containing S; it is denoted by $[S]^*$. Clearly, $[S]^*$ is generated, as an ideal, by the set $\{\gamma(a) \,|\, a \in S,\, \gamma \in \Gamma\})$. If the set S is finite, $S = \{a_1, \ldots, a_r\}$, we say that the σ-ideal $J = [S]^*$ is finitely generated (we write this as $J = [a_1, \ldots, a_r]^*$) and call $a_1, \ldots, a_r$ σ^*-generators of J.

If R_0 is a σ^*-subring of R and $S \subseteq R$, then the smallest σ^*-subring of R containing R_0 and S is denoted by $R_0\{S\}^*$ (if S is finite, $S = \{\eta_1, \ldots, \eta_n\}$, we write $R_0\{\eta_1, \ldots, \eta_n\}^*$). As a ring, it is generated over R_0 by the set $\{\gamma s \,|\, \gamma \in \Gamma,\, s \in S\}$. (Here and below we frequently write γs for $\gamma(s)$.)

A ring homomorphism of σ-rings $\phi : R \longrightarrow S$ is called a *difference* (or σ-) *homomorphism* if $\phi(\alpha a) = \alpha\phi(a)$ for any $\alpha \in \sigma$, $a \in R$ (of course, if R and S are inversive, the equality holds for every $\alpha \in \sigma^*$). It is easy to see that the kernel of such a mapping is a σ^*-ideal of R.

If K is a σ^*-subfield of a σ^*-field L and $S \subseteq L$, then the smallest σ^*-subfield of L containing K and S is denoted by $K\langle S\rangle^*$. If the set S is finite, $S = \{\eta_1, \ldots, \eta_n\}$, then $K\langle S\rangle^*$ is written as $K\langle \eta_1, \ldots, \eta_n\rangle^*$ and is said to be a finitely generated inversive difference (or σ^*-) extension of K with the set of σ^*-generators $\{\eta_1, \ldots, \eta_n\}$. It is easy to see that the field $K\langle \eta_1, \ldots, \eta_n\rangle^*$ coincides with the field $K(\{\gamma\eta_i \,|\, \gamma \in \Gamma,\, 1 \leq i \leq n\})$.

If R is an inversive difference (σ^*-) ring and $Y = \{y_1, \ldots, y_n\}$ is a finite set of symbols, then one can consider a countable set of symbols $\Gamma Y = \{\gamma y_j | \gamma \in \Gamma, 1 \leq j \leq n\}$ and the polynomial ring $R[\Gamma Y]$ in the set of indeterminates ΓY over R (the elements of ΓY will be called *terms*). This polynomial ring is naturally viewed as a σ^*-ring where the action of the translations of σ^* on R is extended to $R[\Gamma Y]$ by setting $\alpha(\gamma y_j) = (\alpha\gamma)y_j$ for any $\alpha \in \sigma^*$, $\gamma \in \Gamma$, $1 \leq j \leq n$. This σ^*-ring (that contains R as its σ^*-subring) is denoted by $R\{y_1, \ldots, y_n\}^*$; it is called the *ring of inversive difference* (or σ^*-) *polynomials* in the set of inversive difference (σ^*-) indeterminates $y_1, \ldots, y_n$ over R. Elements of $R\{y_1, \ldots, y_n\}^*$ are called inversive difference (or σ^*-) polynomials. If R is a σ^*-subring of a σ^*-ring S, $f \in R\{y_1, \ldots, y_n\}^*$ and $\eta = (\eta_1, \ldots, \eta_n) \in S^n$, then $f(\eta)$ denotes the result of the replacement of every entry γy_i in f by $\gamma\eta_i$ ($\gamma \in \Gamma$, $1 \leq i \leq n$).

Let R be an inversive difference (σ^*-) ring and $U = \left\{u^{(\lambda)} | \lambda \in \Lambda\right\}$ a family of elements of some σ^*-overring of R. We say that the family U is *transformally* (or *σ-algebraically) dependent* over R, if the family

$$U_\sigma^* = \left\{\gamma u^{(\lambda)} \,|\, \gamma \in \Gamma,\, \lambda \in \Lambda\right\}$$

is algebraically dependent over R (that is, there exist elements $v^{(1)}, \dots, v^{(k)} \in U_\sigma^*$ and a nonzero polynomial f in k variables with coefficients in R such that $f\big(v^{(1)}, \dots, v^{(k)}\big) = 0$). Otherwise, the family U is said to be *transformally* (or *σ-algebraically) independent* over R.

If K is an inversive difference (σ^*-) field and L a σ^*-field extension of K, then a set $B \subseteq L$ is said to be a *difference* (or σ-) *transcendence basis* of L over K if B is σ-algebraically independent over K and every element $a \in L$ is σ-algebraic over $K\langle B\rangle^*$ (that is, the set $\{\gamma a \,|\, \gamma \in \Gamma\}$ is algebraically dependent over $K\langle B\rangle^*$). If L is a finitely generated σ^*-field extension of K, then all σ-transcendence bases of L over K are finite and have the same number of elements (see [9, Sect. 4.1]). This number is called the *difference* (or σ-) *transcendence degree* of L over K (or the σ-transcendence degree of the extension L/K); it is denoted by $\sigma\text{-trdeg}_K L$.

3 Dimension Quasi-polynomials of Subsets of $\mathbb{N}^m$ and $\mathbb{Z}^m$

A function $f : \mathbb{Z} \to \mathbb{Q}$ is called a *quasi-polynomial* of period q if there exist q polynomials $g_i(x) \in \mathbb{Q}[x]$ $(0 \le i \le q-1)$ such that $f(n) = g_i(n)$ whenever $n \in \mathbb{Z}$ and $n \equiv i\,(mod\, q)$.

An equivalent way of introducing quasi-polynomials is as follows.

A *rational periodic number* $U(n)$ is a function $U : \mathbb{Z} \to \mathbb{Q}$ with the property that there exists (a period) $q \in \mathbb{N}$ such that

$$U(n) = U(n') \text{ whenever } n \equiv n'\,(mod\, q).$$

A rational periodic number is represented by a list of q its possible values as follows:

$$U(n) = [a_0, \dots, a_{q-1}]_n.$$

$(U(n) = a_i$ $(0 \le i \le q-1)$ whenever $n \equiv i\,(mod\, q))$.

For example, $U(n) = \left[\frac{2}{3}, \frac{1}{4}, 5\right]_n$ is a periodic number with period 3 such that $U(n) = \frac{2}{3}$ if $n \equiv 0\,(mod\, 3)$, $U(n) = \frac{1}{4}$ if $n \equiv 1\,(mod\, 3)$, and $U(n) = 5$ if $n \equiv 2\,(mod\, 3)$.

With the above notation, a *quasi-polynomial of degree d* is defined as a function $f : \mathbb{Z} \to \mathbb{Q}$ such that

$$f(n) = c_d(n)n^d + \cdots + c_1(n)n + c_0(n) \quad (n \in \mathbb{Z})$$

where $c_i(n)$'s are rational periodic numbers and $c_d(n) \neq 0$ for at least one $n \in \mathbb{Z}$.

One of the main applications of quasi-polynomials is their application to the problem of counting integer points in rational polytopes. Recall that a *rational polytope* in $\mathbb{R}^d$ is the convex hull of finitely many points (vertices) in $\mathbb{Q}^d$ (d is a positive integer). Equivalently, a rational polytope $P \subseteq \mathbb{R}^d$ is the set of solutions of a finite system of linear inequalities $A\mathbf{x} \le \mathbf{b}$, where A is an $m \times d$-matrix

with integer entries ($m \in \mathbb{Z}$, $m \geq 1$) and $\mathbf{b} \in \mathbb{Z}^m$, provided that the solution set is bounded. A rational polytope is said to be a *lattice* one if all its vertices are integer points (that is, points with integer coordinates).

Let $P \subseteq \mathbb{R}^d$ be a rational polytope. (We assume that P has dimension d, that is, P is not contained in a proper affine subspace of $\mathbb{R}^d$.) Then a polytope

$$rP = \{r\mathbf{x} \,|\, \mathbf{x} \in P\}$$

($r \in \mathbb{N}$, $n \geq 1$) is called the rth *dilate* of P. Clearly, if $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are all vertices of P, then rP is the convex hull of $r\mathbf{v}_1, \ldots, r\mathbf{v}_k$.

Given a rational polytope P, let $L(P, r)$ denote the number of integer points in rP (in other words, $L(P, r) = \mathrm{Card}(rP \cap \mathbb{Z}^d)$). The following result is due to Ehrhart, see [5].

Theorem 1. *Let $P \subseteq \mathbb{R}^d$ be a rational polytope of dimension d. Then there exists a quasi-polynomial $\phi_P(r)$ of degree d such that*

(i) *$\phi_P(r) = L(P, r)$ for all $r \in \mathbb{N}$.*
(ii) *The leading coefficient of $\phi_P(r)$ is a constant that is equal to the Euclidean volume of the polytope P.*
(iii) *The minimum period of $\phi_P(r)$ (that is, the least common multiple of the minimum periods of its coefficients) is a divisor of the number $\mathcal{D}(P) = \min\{n \in \mathbb{N} \,|\, nP$ is a lattice polytope$\}$.*
(iv) *If P is a lattice polytope, then $\phi_P(r)$ is a polynomial of r with rational coefficients.*

The main tools for computation of Ehrhart quasi-polynomials are Alexander Barvinok's polynomial time algorithm and its modifications, see [1–3]. In some cases, however, the Ehrhart quasi-polynomial can be found directly from the Ehrhart's theorem by evaluating the periodic coefficients of the quasi-polynomial (by computing $L(P, r)$ for the first several values of $r \in \mathbb{N}$ and then solving the corresponding system of linear equations, see [12, Example 1]).

Let $w_1, \ldots, w_m$ be fixed positive integers ($m > 0$) and $a = (a_1, \ldots, a_m) \in \mathbb{N}^m$. Then the number

$$\mathrm{ord}_w\, a = w_1 a_1 + \cdots + w_m a_m$$

is called the *order of a with respect to the weights $w_1, \ldots, w_m$*. If the weights are fixed, $\mathrm{ord}_w\, a$ is simply called the *order of a*.

In what follows, $\lambda_w^{(m)}(t)$ denotes the Ehrhart quasi-polynomial that describes the number of integer points in the *conic polytope*

$$P_t = \{(x_1, \ldots, x_m) \in \mathbb{R}^m \mid \sum_{i=1}^{m} w_i x_i \leq t,\, x_j \geq 0\, (1 \leq j \leq m)\}.$$

It follows from the Ehrhart's Theorem that $\lambda_w^{(m)}(t)$ is a quasi-polynomial of degree m whose leading coefficient is $\dfrac{1}{m! w_1 \ldots w_m}$. A polynomial time algorithm

for computing $\lambda_w^{(m)}(t)$ can be found, for example, in [3]. We will illustrate the computation of such a quasi-polynomial with the use of the above-mentioned method based on the Ehrhart's theorem.

Example 1. Consider a conic polytope

$$P = \{(x_1, x_2) \in \mathbb{R}^2 \,|\, x_1 \geq 0,\ x_2 \geq 0,\ 2x_1 + 3x_2 \leq 1\}$$

whose rth dilate is

$$rP = \{(x_1, x_2) \in \mathbb{R}^2 \,|\, x_1 \geq 0,\ x_2 \geq 0,\ 2x_1 + 3x_2 \leq r\}.$$

By the Ehrhart's theorem,

$$\phi_P(r) = L(P, r) = \frac{1}{12}r^2 + [a_0, a_1, a_2, a_3, a_4, a_5]_r r + [b_0, b_1, b_2, b_3, b_4, b_5]_r$$

where $a_i, b_i \in \mathbb{Q}$ $(0 \leq i \leq 5)$. The direct computation of the number of integer points in the first eleven dilated polytopes rP gives

$\phi_P(0) = b_0 = 1,\ \phi_P(1) = \frac{1}{12} + a_1 + b_1 = 1,\ \phi_P(2) = \frac{1}{3} + 2a_2 + b_2 = 2,$
$\phi_P(3) = \frac{3}{4} + 3a_3 + b_3 = 3,\ \phi_P(4) = \frac{4}{3} + 4a_4 + b_4 = 4,\ \phi_P(5) = \frac{25}{12} + 5a_5 + b_5 = 5,$
$\phi_P(6) = 3 + 6a_0 + b_0 = 7;\ \phi_P(7) = \frac{49}{12} + 7a_1 + b_1 = 8,\ \phi_P(8) = \frac{16}{3} + 8a_2 + b_2 = 10,$
$\phi_P(9) = \frac{27}{4} + 9a_3 + b_3 = 12,\ \phi_P(10) = \frac{25}{3} + 10a_4 + b_4 = 14,$
$\phi_P(11) = \frac{121}{12} + 11a_5 + b_5 = 16$. Solving this system we obtain that $a_i = \frac{1}{2}$ $(0 \leq i \leq 5)$, $b_0 = 1$, $b_1 = \frac{5}{12}$, $b_2 = \frac{2}{3}$, $b_3 = \frac{3}{4}$, $b_4 = \frac{2}{3}$, and $b_5 = \frac{5}{12}$, so that

$$\phi_P(r) = \frac{1}{12}r^2 + \frac{3}{4}r + \left[1, \frac{5}{12}, \frac{2}{3}, \frac{3}{4}, \frac{2}{3}, \frac{5}{12}\right]_r.$$

Remark 1. In what follows we will also use Ehrhart quasi-polynomials that describe (for sufficiently large $r \in \mathbb{N}$) the numbers of integer points $(x_1, \ldots, x_m) \in \mathbb{R}^m$ satisfying the inequality $|x_1| + \cdots + |x_m| \leq r$. It is easy to see that such a quasi-polynomial, denoted by $\mu_w^{(m)}(t)$, is an alternating sum of quasi-polynomials of the form $\lambda_w^{(k)}(t)$ that contains 2^m terms $\lambda_w^{(m)}(t)$ and the other quasi-polynomials $\lambda_w^{(k)}(t)$ in this sum have $k < m$. It follows that

$$\mu_w^{(m)}(t) = \frac{2^m}{m! w_1 \ldots w_m} t^m + \text{ terms of degree less than } m.$$

For any $A \subseteq \mathbb{N}^m$, $r \in \mathbb{N}$ (and the fixed weight vector $w = (w_1, \ldots, w_m)$), we set

$$A^{(w)}(r) = \{a = (a_1, \ldots, a_m) \in A \,|\, \operatorname{ord}_w a \leq r\}.$$

Furthermore, $V_A^{(w)}$ will denote the set of all m-tuples $v = (v_1, \ldots, v_m) \in \mathbb{N}^m$ that are not greater than or equal to any m-tuple from A with respect to the product order $\leq_P$. In other words, an element $v = (v_1, \ldots, v_m) \in \mathbb{N}^m$ lies in $V_A^{(w)}$ if and only if for any element $(a_1, \ldots, a_m) \in A$ there exists $i \in \mathbb{N}, 1 \leq i \leq m$, such that $a_i > v_i$.

The following theorem proved in [12] generalizes E. Kolchin's result on dimension polynomials of subsets of $\mathbb{N}^m$ (see [6, Chap. 0, Lemma 16]).

Theorem 2. *With the above conventions, for any set $A \subseteq \mathbb{N}^m$, there exists a quasi-polynomial $\chi_A^{(w)}(t)$ in one variable t such that*

(i) $\chi_A^{(w)}(r) = \operatorname{Card} V_A^{(w)}(r)$ *for all sufficiently large* $r \in \mathbb{N}$.
(ii) $\deg \chi_A^{(w)}(t) \leq m$.
(iii) $\deg \chi_A^{(w)}(t) = m$ *if and only if* $A = \emptyset$. *In this case* $\chi_A^{(w)}(t) = \lambda_w^{(m)}(t)$.
(iv) $\chi_A^{(w)}(t) = 0$ *if and only if* $(0, \ldots, 0) \in A$.
(v) *Let* $A = \{a^{(1)}, \ldots, a^{(d)}\}$ *be a finite subset of* $\mathbb{N}^m$ *and let* $a^{(i)} = (a_{i1}, \ldots, a_{im})$ *for* $i = 1, \ldots, d$. *For any* $l = 0, \ldots, d$, *let* $\Delta(l, d)$ *denote the set of all* l*-element subsets of* $\{1, \ldots, d\}$ *(*$\Delta(0, d) = \emptyset$*) and for any set* $\epsilon = \{a^{(i_1)}, \ldots, a^{(i_l)}\} \in \Delta(l, d)$ *(*$1 \leq i_1 < \cdots < i_l \leq d$*), let* $c_{\epsilon j} = \max\{a_{\nu j} \mid \nu = i_1, \ldots, i_l\}$ *(the maximal* j*th coordinate of elements of* ϵ*). Furthermore, let* $c_\epsilon = (c_{\epsilon 1}, \ldots, c_{\epsilon m})$. *Then*

$$\chi_A^{(w)}(t) = \sum_{l=0}^{d} (-1)^l \sum_{\epsilon \in \Delta(l,d)} \lambda_w^{(m)}(t - \operatorname{ord}_w c_\epsilon). \tag{1}$$

The quasi-polynomial $\chi_A^{(w)}(t)$ whose existence is established by Theorem 2 is called the *dimension quasi-polynomial of the set* $A \subseteq \mathbb{N}^m$ *associated with the weight vector* $(w_1, \ldots, w_m)$. An example of computation of such a quasi-polynomial can be found in [12, Example 2].

Note that if $w_1 = \cdots = w_m = 1$, then $\chi_A^{(w)}(t)$ is the dimension polynomial of the subset A of $\mathbb{N}^m$ introduced in [6, Chap. 0, Lemma 16]. Some properties of such polynomials and methods of their computation were obtained in [7, Chap. 2]. If $w_1 = \cdots = w_m = w$ and $w > 1$, then it is easy to see that $\lambda_w^{(m)}(t)$ describes the number of integer points in the conic polytope $\{(x_1, \ldots, x_m) \in \mathbb{R}^m \mid \sum_{i=1}^m x_i \leq \lfloor \frac{t}{w} \rfloor\}$ ($\lfloor a \rfloor$ denotes the greatest integer not exceeding a). Therefore, $\lambda_w^{(m)}(t) = \binom{\lfloor \frac{t}{w} \rfloor + m}{m}$. Clearly, w is the minimum period of this quasi-polynomial ($\lambda_w^{(m)}(t) = \binom{\frac{t-i}{w} + m}{m}$ whenever $t \in \mathbb{Z}$ and $t \equiv i \,(\mathrm{mod}\, w)$, $0 \leq i \leq w - 1$). It follows that in this case w is a period of the dimension quasi-polynomial $\chi_A^{(w)}(t)$, since every term in the sum (1) is of the form $\binom{\lfloor \frac{t - \operatorname{ord}_w c_\epsilon}{w} \rfloor + m}{m}$.

Now we extend previous considerations to subsets of $\mathbb{Z}^m$ ($m \in \mathbb{N}$, $m \geq 1$). We fix positive integers $w_1, \ldots, w_m$ ("weights") and define the *order* of an m-tuple $a = (a_1, \ldots, a_m) \in \mathbb{Z}^m$ (with respect to the given weights) as

$$\operatorname{ord}_w a = w_1|a_1| + \cdots + w_m|a_m|.$$

If $A \subseteq \mathbb{Z}^m$ and $r \in \mathbb{N}$, we set $A(r) = \{a \in A \mid \operatorname{ord}_w a \leq r\}$.

In what follows the set $\mathbb{Z}^m$ will be considered as the union

$$\mathbb{Z}^m = \bigcup_{1 \leq j \leq 2^m} \mathbb{Z}_j^{(m)} \tag{2}$$

where $\mathbb{Z}_1^{(m)}, \ldots, \mathbb{Z}_{2^m}^{(m)}$ are all distinct Cartesian products of m factors each of which is either $\mathbb{Z}_-$ or $\mathbb{N}$. We assume that $\mathbb{Z}_1^{(m)} = \mathbb{N}^m$ and call a set $\mathbb{Z}_j^{(m)}$ $(1 \leq j \leq 2^m)$ an *orthant* of $\mathbb{Z}^m$.

The set $\mathbb{Z}^m$ will be treated as a partially ordered set with respect to the order $\trianglelefteq$ defined as follows: $(x_1, \ldots, x_m) \trianglelefteq (y_1, \ldots, y_m)$ if and only if $(x_1, \ldots, x_m)$ and $(y_1, \ldots, y_m)$ belong to the same orthant of $\mathbb{Z}^m$ and $|x_i| \leq |y_i|$ for $i = 1, \ldots, m$. For any set $B \subseteq \mathbb{Z}^m$, $\mathcal{V}_B$ will denote the set of all m-tuples in $\mathbb{Z}^m$ that exceed no element of B with respect to $\trianglelefteq$. Furthermore, we define a mapping $\rho : \mathbb{Z}^m \to \mathbb{N}^{2m}$ such that

$$\rho(z_1, \ldots, z_m) = (\max\{z_1, 0\}, \ldots, \max\{z_m, 0\}, \max\{-z_1, 0\}, \ldots, \max\{-z_m, 0\}).$$

The proof of the following theorem can be obtained by mimicking the proof of the first three parts of Theorem 2.5.5 of [7] (with the use of Theorem 2 instead of Theorem 2.2.5 of [7]).

Theorem 3. *Let $\mathcal{A}$ be a subset of $\mathbb{Z}^m$. Then there exists a quasi-polynomial $\phi_{\mathcal{A}}^{(w)}(t)$ with the following properties.*

(i) $\phi_A^{(w)}(r) = \operatorname{Card} \mathcal{V}_{\mathcal{A}}(r)$ *for all sufficiently large $r \in \mathbb{N}$;*

(ii) $\deg \phi_{\mathcal{A}}^{(w)} \leq m$.

(iii) *Let $\rho(\mathcal{A}) = A$ $(A \subseteq \mathbb{N}^{2m})$ and let b_i $(1 \leq i \leq m)$ be a $2m$-tuple in $\mathbb{N}^{2m}$ whose ith and $(m+i)$th coordinates are 1 and all other coordinates are zeros. Then for all $r \in \mathbb{N}$, $\phi_{\mathcal{A}}^{(w)}(r) = \chi_B^{(w)}(r)$ where $B = A \cup \{b_1\} \cup \cdots \cup \{b_m\}$ and $\chi_B^{(w)}(t)$ is the dimension quasi-polynomial of the set $B \subseteq \mathbb{N}^{2m}$ associated with the weight vector $(w_1, \ldots, w_m, w_1, \ldots, w_m)$.*

The quasi-polynomial $\phi_{\mathcal{A}}^{(w)}(t)$ is called the *dimension polynomial of the set $\mathcal{A} \subseteq \mathbb{Z}^m$ associated with the weight vector $(w_1, \ldots, w_m)$*.

4 The Main Theorem

Let K be an inversive difference field of characteristic zero with a basic set of translations $\sigma = \{\alpha_1, \ldots, \alpha_m\}$ that are assigned positive integer weights $w_1, \ldots, w_m$, respectively. As before, let Γ denote the free commutative group generated by the set σ. For any element $\gamma = \alpha_1^{k_1} \ldots \alpha_m^{k_m} \in \Gamma$, the number

$$\operatorname{ord}_w \gamma = \sum_{i=1}^{m} w_i |k_i|$$

will be called the *order* of γ. Furthermore, for any $r \in \mathbb{N}$, we set

$$\Gamma_w(r) = \{\gamma \in \Gamma \mid \operatorname{ord}_w \gamma \leq r\}.$$

The following theorem establishes the existence of a dimension quasi-polynomial associated with a finitely generated inversive difference field extension with weighted basic translations.

Theorem 4. *With the above notation, let $L = K\langle\eta_1, \ldots, \eta_n\rangle^*$ be a σ^*-field extension of K generated by a finite set $\eta = \{\eta_1, \ldots, \eta_n\}$. For any $r \in \mathbb{N}$, let $L_r = K(\{\gamma\eta_i \,|\gamma \in \Gamma_w(r), 1 \leq i \leq n\})$. Then there exists a quasi-polynomial $\Psi^{(w)}_{\eta|K}(t)$ such that*

(i) *$\Psi^{(w)}_{\eta|K}(r) = \mathrm{trdeg}_K L_r$ for all sufficiently large $r \in \mathbb{N}$.*

(ii) *$\deg \Psi^{(w)}_{\eta|K} \leq m$.*

(iii) *$\Psi^{(w)}_{\eta|K}$ is an alternating sum of Ehrhart quasi-polynomials associated with rational conic polytopes in $\mathbb{N}^m$.*

(iv) *The degree and leading coefficient of the quasi-polynomial $\Psi^{(w)}_{\eta|K}$ are constants that do not depend on the set of σ^*-generators η of the extension* L/K*. Furthermore, the coefficient of t^m in $\Psi^{(w)}_{\eta|K}$ can be represented as $\dfrac{a2^m}{m!w_1 \ldots w_m}$ where a is equal to the σ-transcendence degree of* L/K*.*

The quasi-polynomial $\Psi^{(w)}_{\eta|K}(t)$ is called the *σ^*-dimension quasi-polynomial* of the σ^*-field extension L/K associated with the system of σ^*-generators η.

In order to prove Theorem 4 we need some results on reduction and autoreduced sets in the ring of inversive difference polynomials $K\{y_1, \ldots, y_n\}^*$. In what follows we will consider $\mathbb{Z}^m$ as the union (2) of 2^m orthants $\mathbb{Z}^{(m)}_j$ $(1 \leq j \leq 2^m)$, and the group Γ will be considered as a union $\Gamma = \bigcup_{j=1}^{2^m} \Gamma_j$ where $\Gamma_j = \{\alpha_1^{k_1} \ldots \alpha_m^{k_m} \,|\, (k_1, \ldots, k_m) \in \mathbb{Z}^{(m)}_j\}$ $(1 \leq j \leq 2^m)$.

Let $K\{y_1, \ldots, y_n\}^*$ be an algebra of σ^*-polynomials in σ^*-indeterminates $y_1, \ldots, y_n$ over K; as before, ΓY will denote the set of terms $\{\gamma y_i | \gamma \in \Gamma, 1 \leq i \leq n\}$. By the order of a term $u = \gamma y_j$ we mean the order of the element $\gamma \in \Gamma$. Setting $(\Gamma Y)_j = \{\gamma y_i \,|\, \gamma \in \Gamma_j, 1 \leq i \leq n\}$ $(j = 1, \ldots, 2^m)$ we obtain a representation of the set of terms as a union $\Gamma Y = \bigcup_{j=1}^{2^m} (\Gamma Y)_j$.

Definition 1. *A term $v \in \Gamma Y$ is called a transform of a term $u \in Y$ if and only if u and v belong to the same set $(\Gamma Y)_j$ $(1 \leq j \leq 2^m)$ and $v = \gamma u$ for some $\gamma \in \Gamma_j$ (in particular, u and v involve the same σ^*-indeterminate y_i, $1 \leq i \leq n$). If $\gamma \neq 1$, v is said to be a proper transform of u.*

Definition 2. *A well-ordering of the set ΓY is called a ranking of the family of σ^*-indeterminates $y_1, \ldots, y_n$ (or a ranking of the set of terms ΓY) if it satisfies the following conditions. (We use the standard symbol $\leq$ for the ranking; it will be always clear what order is denoted by this symbol.)*

(i) *If $u \in (\Gamma Y)_j$ and $\gamma \in \Gamma_j$ $(1 \leq j \leq 2^m)$, then $u \leq \gamma u$.*

(ii) *If $u, v \in (\Gamma Y)_j$ $(1 \leq j \leq 2^m)$, $u \leq v$ and $\gamma \in \Gamma_j$, then $\gamma u \leq \gamma v$.*

A ranking of the σ^*-indeterminates $y_1, \dots, y_n$ is called *orderly* if for any $j = 1, \dots, 2^m$ and for any two terms $u, v \in \Gamma Y$, the inequality $\operatorname{ord}_w u < \operatorname{ord}_w v$ implies that $u < v$ (as usual, $v < w$ means $v \leq w$ and $v \neq w$). As an example of an orderly ranking one can consider the *standard ranking* defined as follows: $u = \alpha_1^{k_1} \dots \alpha_m^{k_m} y_i \leq v = \alpha_1^{l_1} \dots \alpha_m^{l_m} y_j$ if and only if the $(2m+2)$-tuple $(\sum_{\nu=1}^{m} |k_\nu|, |k_1|, \dots, |k_m|, k_1, \dots, k_m, i)$ is less than or equal to the $(2m+2)$-tuple $(\sum_{\nu=1}^{m} |l_\nu|, |l_1|, \dots, |l_m|, l_1, \dots, l_m, j)$ with respect to the lexicographic order on $\mathbb{Z}^{2m+2}$.

In what follows, we assume that an orderly ranking $\leq$ of the set of σ^*-indeterminates $y_1, \dots, y_n$ is fixed. If $A \in K\{y_1, \dots, y_n\}^*$, then the greatest (with respect to $\leq$) term from ΓY that appears in A is called the *leader* of A; it is denoted by u_A. If $u = u_A$ and $d = \deg_u A$, then the σ^*-polynomial A can be written as $A = I_d u^d + I_{d-1} u^{d-1} + \dots + I_0$ where I_k $(0 \leq k \leq d)$ do not contain u. The σ^*-polynomial I_d is called the *initial* of A; it is denoted by I_A.

The ranking of the set of σ^*-indeterminates $y_1, \dots, y_n$ generates the following relation on $K\{y_1, \dots, y_n\}^*$. If A and B are two σ^*-polynomials, then A is said to have rank less than B (we write $A < B$) if either $A \in K, B \notin K$ or $A, B \in K\{y_1, \dots, y_n\}^* \setminus K$ and $u_A < u_B$, or $u_A = u_B = u$ and $\deg_u A < \deg_u B$. If $u_A = u_B = u$ and $\deg_u A = \deg_u B$, we say that A and B are of the same rank and write $\operatorname{rk} A = \operatorname{rk} B$.

Let $A, B \in K\{y_1, \dots, y_n\}^*$. The σ^*-polynomial A is said to be *reduced* with respect to B if A does not contain any power of a transform γu_B $(\gamma \in \Gamma)$ whose exponent is greater than or equal to $\deg_{u_B} B$. If $\mathcal{A} \subseteq K\{y_1, \dots, y_n\} \setminus K$, then a σ^*-polynomial $A \in K\{y_1, \dots, y_n\}^*$, is said to be reduced with respect to $\mathcal{A}$ if A is reduced with respect to every element of the set $\mathcal{A}$.

A set $\mathcal{A} \subseteq K\{y_1, \dots, y_n\}^*$ is said to be *autoreduced* if either it is empty or $\mathcal{A} \bigcap K = \emptyset$ and every element of $\mathcal{A}$ is reduced with respect to all other elements of $\mathcal{A}$. As it is shown in [9, Sect. 2.4], every autoreduced set is finite, distinct elements of an autoreduced set have distinct leaders, and one has the following result.

Theorem 5. *Let $\mathcal{A} = \{A_1, \dots, A_p\}$ be an autoreduced subset in $K\{y_1, \dots, y_n\}^*$ and let $D \in K\{y_1, \dots, y_n\}^*$. Furthermore, let $I(\mathcal{A})$ denote the set of all σ^*-polynomials $B \in K\{y_1, \dots, y_n\}$ such that either $B = 1$ or B is a product of finitely many polynomials of the form $\gamma(I_{A_i})$ where $\gamma \in \Gamma$, $i = 1, \dots, p$. Then there exist σ^*-polynomials $J \in I(\mathcal{A})$ and $E \in K\{y_1, \dots, y_n\}$ such that E is reduced with respect to $\mathcal{A}$ and $JD \equiv E (\operatorname{mod} [\mathcal{A}])$.*

The transition from a σ^*-polynomial D to the σ^*-polynomial E (called a *remainder* of D with respect to $\mathcal{A}$) *can be realized by the algorithm described in* [9, p. 131]. *In this case we say that D reduces to E modulo $\mathcal{A}$.*

In what follows the elements of an autoreduced set will be always written in the order of increasing rank. If $\mathcal{A} = \{A_1, \dots, A_p\}$ and $\mathcal{B} = \{B_1, \dots, B_q\}$ are two

autoreduced sets of σ^*-polynomials, we say that $\mathcal{A}$ has lower rank than $\mathcal{B}$ and write $\operatorname{rk}\mathcal{A} < \operatorname{rk}\mathcal{B}$ if either there exists $k \in \mathbf{N}$, $1 \leq k \leq \min\{p, q\}$, such that $\operatorname{rk} A_i = \operatorname{rk} B_i$ for $i = 1, \ldots, k-1$ and $A_k < B_k$, or $p > q$ and $\operatorname{rk} A_i = \operatorname{rk} B_i$ for $i = 1, \ldots, q$.

Mimicking the proof of [6, Chap. I, Proposition 3] we obtain that every nonempty family of autoreduced subsets contains an autoreduced set of lowest rank. In particular, if $\emptyset \neq J \subseteq K\{y_1, \ldots, y_n\}^*$, then the set J contains an autoreduced set of lowest rank called a *characteristic set* of J. We will need the following properties of characteristic sets that follow from Theorem 5 (see [9, Proposition 2.4.4]).

Proposition 1. *Let K be an inversive difference (σ^*-) field, J a σ^*-ideal of the algebra of σ^*-polynomials $K\{y_1, \ldots, y_s\}^*$, and $\mathcal{A}$ a characteristic set of J. Then*

(i) *The ideal J does not contain nonzero σ^*-polynomials reduced with respect to $\mathcal{A}$. In particular, if $A \in \mathcal{A}$, then $I_A \notin J$.*

(ii) *If J is a prime σ^*-ideal, then $J = [\mathcal{A}] : \Upsilon(\mathcal{A})$ where $\Upsilon(\mathcal{A})$ denotes the set of all finite products of elements of the form $\gamma(I_A)$ ($\gamma \in \Gamma, A \in \mathcal{A}$).*

Proof of Theorem 4

Let P be the defining σ^*-ideal of the σ^*-field extension $L = K\langle \eta_1, \ldots, \eta_n \rangle^*$, that is $P = \operatorname{Ker}(K\{y_1, \ldots, y_n\}^* \to L)$, $y_i \mapsto \eta_i$. Let $\mathcal{A} = \{A_1, \ldots, A_d\}$ be a characteristic set of P, let u_i denote the leader of A_i $(1 \leq i \leq d)$ and for every $j = 1, \ldots, n$, let

$$E_j = \{(k_1, \ldots, k_m) \in \mathbb{Z}^m | \alpha_1^{k_1} \ldots \alpha_m^{k_m} y_j \text{ is a leader of a } \sigma\text{-polynomial in } \mathcal{A}\}.$$

Let $V = \{u \in \Gamma Y \mid u \text{ is not a transform of any } u_i \ (1 \leq i \leq s)\}$ and for every $r \in \mathbb{N}$, let $V(r) = \{u \in V \mid \operatorname{ord}_w u \leq r\}$.

By Proposition 1, the ideal P does not contain non-zero difference polynomials reduced with respect to $\mathcal{A}$. It follows that for every $r \in \mathbb{N}$, the set $V_\eta(r) = \{v(\eta) \mid v \in V(r)\}$ is algebraically independent over K. Indeed, if there exists a nonzero polynomial $B \in K[X_1, \ldots, X_k]$ $(k > 0)$ in k variables over K and elements $v_1, \ldots, v_k \in V_\eta(r)$ such that $B(v_1(\eta), \ldots, v_k(\eta)) = 0$. Then $B(v_1, \ldots, v_k) \in P$ and this σ-polynomial is reduced with respect to $\mathcal{A}$, so we arrive at a contradiction.

If $A_i \in \mathcal{A}$ $(1 \leq i \leq d)$, then $A_i(\eta) = 0$, hence $u_i(\eta)$ is algebraic over the field $K(\{\gamma\eta_j \mid \gamma y_j < u_i \ (\gamma \in \Gamma, 1 \leq j \leq n)\})$. Therefore, any transform θu_i $(\theta \in \Gamma)$ is algebraic over the field $K(\{\gamma\eta_j \mid \gamma y_j < \theta u_i \ (\gamma \in \Gamma, 1 \leq j \leq n)\})$. By induction on the well-ordered set ΓY (and using the fact that if $u, v \in \Gamma Y$ and $u < v$, then $\operatorname{ord}_w u \leq \operatorname{ord}_w v$), we obtain that for every $r \in \mathbb{N}$, the field $L_r = K(\{\gamma\eta_j \mid \gamma \in \Gamma_w(r), 1 \leq j \leq n\})$ is an algebraic extension of the field $K(\{v(\eta) \mid v \in V(r)\})$. It follows that $V_\eta(r)$ is a transcendence basis of L_r over K and $\operatorname{trdeg}_K L_r = \operatorname{Card} V_\eta(r)$.

The number of terms $\alpha_1^{k_1} \ldots \alpha_m^{k_m} y_j$ in $V(r)$ is equal to the number of m-tuples $k = (k_1, \ldots, k_m) \in \mathbb{Z}^m$ such that $\operatorname{ord}_w k \leq r$ and k does not exceed any m-tuple

in E_j with respect to the order $\trianglelefteq$ on $\mathbb{Z}^m$. By Theorem 3, this number is expressed by a quasi-polynomial of degree at most m. Therefore, for all sufficiently large $r \in \mathbb{N}$ we have

$$\operatorname{trdeg}_K L_r = \operatorname{Card} V_\eta(r) = \sum_{j=1}^{n} \phi_{E_j}^{(w)}(r)$$

where $\phi_{E_j}^{(w)}(t)$ is the dimension quasi-polynomial of the set $E_j \subseteq \mathbb{Z}^m$. It follows that the quasi-polynomial

$$\Psi_{\eta|K}^{(w)}(t) = \sum_{j=1}^{n} \phi_{E_j}^{(w)}(t)$$

satisfies the first two conditions of Theorem 4. Since each $\phi_{E_j}^{(w)}(t)$ is equal to a dimension quasi-polynomial $\chi_{B_j}^{(w)}(t)$ of a subset $B_j \subseteq \mathbb{N}^{2m}$ (see part (iii) of Theorem 3) and by Theorem 2 each $\chi_{B_j}^{(w)}(t)$ $(1 \le j \le n)$ is an alternating sum of Ehrhart quasi-polynomials associated with conic polytopes, $\Psi_{\eta|K}^{(w)}(t)$ has a similar representation and satisfies condition (iii) of Theorem 4.

If $\zeta = (\zeta_1, \dots, \zeta_k)$ is another system of σ-generators of the extension L/K, so that $L = K\langle \eta_1, \dots, \eta_n \rangle^* = K\langle \zeta_1, \dots, \zeta_k \rangle^*$, then there exists $q \in \mathbb{N}$ such that $\eta_1, \dots, \eta_n \in K(\bigcup_{i=1}^{k} \Gamma_w(q)\zeta_i)$ and $\zeta_1, \dots, \zeta_k \in K(\bigcup_{i=1}^{n} \Gamma_w(q)\eta_i)$. Therefore, for all sufficiently large $r \in \mathbb{N}$ (namely, for all $r \ge q$), one has

$$K(\bigcup_{i=1}^{n} \Gamma_w(r)\eta_i) \subseteq K(\bigcup_{i=1}^{k} \Gamma_w(r+q)\zeta_i) \quad \text{and} \quad K(\bigcup_{i=1}^{k} \Gamma_w(r)\zeta_i) \subseteq K(\bigcup_{i=1}^{n} \Gamma_w(r+q)\eta_i),$$

that is, $\Psi_{\eta|K}^{(w)}(r) \le \Psi_{\zeta|K}^{(w)}(r+q)$ and $\Psi_{\zeta|K}^{(w)}(r) \le \Psi_{\eta|K}^{(w)}(r+q)$. It follows that the quasi-polynomials $\Psi_{\eta|K}^{(w)}(t)$ and $\Psi_{\zeta|K}^{(w)}(t)$ have equal degrees and equals leading coefficients.

In order to prove the last part of statement (iv) of our theorem, note first that if the elements $\eta_1, \dots, \eta_n$ are σ-algebraically independent over K, then one has

$$\Psi_{\eta|K}^{(w)}(t) = n\mu_w^{(m)}(t)$$

(see Remark 1). Indeed, if $r \in \mathbb{N}$, and $\Omega_i(r) = \{\omega = \alpha_1^{k_1} \dots \alpha_m^{k_m} \eta_i \,|\, k_i \in \mathbb{Z},\ \operatorname{ord}_w \xi \le r\}$ for $i = 1, \dots, n$, then $\bigcup_{i=1}^{n} \Omega_i(r)$ is a transcendence basis of the field extension $K(\bigcup_{i=1}^{n} \Gamma_w(r)\eta_i)/K$ and the number of elements of this basis is equal to $n \operatorname{Card}\{(k_1, \dots, k_m) \in \mathbb{Z}^m \mid \sum_{j=1}^{m} w_j|k_j| \le r\} = n\mu_w^{(m)}(r)$. Now one can mimic the proof of the last part of [12, Theorem 4] (with the use of quasi-polynomials $\mu_w^{(m)}(t)$ instead of $\lambda_w^{(m)}(t)$) to obtain that the coefficient of t^m in $\Psi_{\eta|K}^{(w)}$ can be represented as $\dfrac{a2^m}{m! w_1 \dots w_m}$ where a is equal to the σ-transcendence degree of L/K. $\square$

Theorem 4 allows one to assign a quasi-polynomial to a system of algebraic difference equations with weighted basic translations

$$f_i(y_1, \ldots, y_n) = 0 \qquad (i = 1, \ldots, p) \tag{3}$$

$(f_i, \in R = K\{y_1, \ldots, y_n\}$ for $i = 1, \ldots, p)$ such that the σ^*-ideal P of R generated by the σ^*-polynomials $f_1, \ldots, f_p$ is prime (e.g. to a system of linear difference equations). Systems of this form arise, in particular, as finite difference approximations of systems of PDEs with weighted derivatives (see, for example, [14,15]).

Treating the quotient field $L = \mathrm{qf}(R/P)$ as a finitely generated σ^*-field extension of K, $L = K\langle \eta_1, \ldots, \eta_n \rangle^*$ where η_i is the canonical image of y_i in R/P, one can consider the σ^*-dimension quasi-polynomial $\Psi^{(w)}(t) = \Psi^{(w)}_{\eta|K}(t)$ associated with this extension. This quasi-polynomial, that is called the σ^*-*dimension quasi-polynomial* of system (3), has a natural interpretation as the Einstein's strength of the system of partial difference equations with weighted translations (see [9, Sect. 7.7]).

Example 2. Let K be an inversive difference field of zero characteristic with a basic set $\sigma = \{\alpha_1, \alpha_2\}$, where α_1 and α_2 are assigned weights 3 and 1, respectively, and let $K\{y\}^*$ be the ring of σ^*-polynomials in one σ^*-indeterminate y over K. Let us consider the σ^*-equation

$$[a_1(\alpha_1 + \alpha_1^{-1} - 2) + a_2(\alpha_2 + \alpha_2^{-1} - 2)]y = 0 \tag{4}$$

where a_1 and a_2 are constants of the field K. As it is shown in [9, Example 7.8.9], the σ^*-polynomials $A = [a_1(\alpha_1 + \alpha_1^{-1} - 2) + a_2(\alpha_2 + \alpha_2^{-1} - 2)]y$, $\alpha_1^{-1}A = [a_1(1 + \alpha_1^{-2} - 2\alpha_1^{-1}) + a_2(\alpha_1^{-1}\alpha_2 + \alpha_1^{-1}\alpha_2^{-1} - 2\alpha_1^{-1})]y$, and $\alpha_1^{-1}\alpha_2^{-1}A = [a_1(\alpha_2^{-1} + \alpha_1^{-2}\alpha_2^{-1} - 2\alpha_1^{-1}\alpha_2^{-1}) + a_2(\alpha_1^{-1} + \alpha_1^{-1}\alpha_2^{-2} - 2\alpha_1^{-1}\alpha_2^{-1})]y$ form a characteristic set of the prime σ^*-ideal $[A]^*$ (clearly, it is irrelevant that all translations in [9] have weight 1). The leaders of these σ^*-polynomials are the terms $\alpha_1 y$, $\alpha_1^{-1}\alpha_2 y$, and $\alpha_1^{-1}\alpha_2^{-2} y$, respectively, so the σ^*-dimension polynomial of the Eq. (4) is equal to the dimension polynomial of the subset $\mathcal{M} = \{(1, 0), (-1, 1), (-1, -2)\}$ of $\mathbb{Z}^2$. In order to find this polynomial one can either represent it as an alternating sum of Ehrhart quasi-polynomials of conic polytopes (using a partition of the set $V_{\mathcal{M}}$ into cones and the principle of inclusion and exclusion, as it is done in [12, Example 2]) or compute the number of integer points of $V_{\mathcal{M}}$ directly (in this case $V_{\mathcal{M}}(r)$ consists of points $(0, i)$ with $-r \leq i \leq r$, points $(j, 0)$ with $-\frac{r}{3} \leq j \leq -1$, and points $(k, -1)$ with $-\frac{r-1}{3} \leq k \leq -1$). We obtain that dimension quasi-polynomial of the Eq. (4) is

$$\Psi(t) = \frac{8}{3}r + \left[0, \frac{1}{3}, -\frac{1}{3}\right]_r.$$

This work was supported by the NSF grant CCF-1714425.

References

1. Barvinok, A.I.: Computing the Ehrhart polynomial of a convex lattice polytope. Discrete Comput. Geom. **12**, 35–48 (1994)
2. Barvinok, A.I., Pommersheim, J.E.: An algorithmic theory of lattice points in polyhedra. In: New Perspectives in Algebraic Combinatorics. Math. Sci. Res. Inst. Publ., vol. 38, pp. 91–147. Cambridge Univ. Press (1999)
3. Barvinok, A.I.: Computing the Ehrhart quasi-polynomial of a rational simplex. Math. Comp. **75**(255), 1449–1466 (2006)
4. Dönch, C.: Standard bases in finitely generated difference-skew-differential modules and their application to dimension polynomials. Ph.D. thesis. Johannes Kepler University Linz, Research Institute for Symbolic Computation (RISC) (2012)
5. Ehrhart, E.: Sur les polyèdres rationnels homothétiques à n dimensions. C. R. Acad. Sci. Paris **254**, 616–618 (1962)
6. Kolchin, E.R.: Differential Algebra and Algebraic Groups. Academic Press, New York (1973)
7. Kondrateva, M.V., Levin, A.B., Mikhalev, A.V., Pankratev, E.V.: Differential and Difference Dimension Polynomials. Kluwer Academic Publishers, Dordrecht (1999)
8. Levin, A.B.: Type and dimension of inversive difference vector spaces and difference algebras. VINITI, Moscow, Russia, no. 1606–82, pp. 1–36 (1982)
9. Levin, A.: Difference Algebra. Springer, New York (2008). https://doi.org/10.1007/978-1-4020-6947-5
10. Levin, A.: Dimension polynomials of intermediate fields of inversive difference field extensions. In: Kotsireas, I.S., Rump, S.M., Yap, C.K. (eds.) MACIS 2015. LNCS, vol. 9582, pp. 362–376. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32859-1_31
11. Levin, A.B.: Dimension polynomials of difference local algebras. Adv. Appl. Math. **72**, 166–174 (2016)
12. Levin, A.B.: Difference dimension quasi-polynomials. Adv. Appl. Math. **89**, 1–17 (2017)
13. Levin, A.B., Mikhalev, A.V.: Type and dimension of finitely generated G-algebras. Contemp. Math. **184**, 275–280 (1995)
14. Shananin, N.A.: On the unique continuation of solutions of differential equations with weighted derivatives. Sb. Math. **191**(3–4), 431–458 (2000)
15. Shananin, N.A.: On the partial quasianalyticity of distribution solutions of weakly nonlinear differential equations with weights assigned to derivatives. Math. Notes **68**(3–4), 519–527 (2000)

Efficient Certification of Numeric Solutions to Eigenproblems

Joris van der Hoeven[1(✉)] and Bernard Mourrain[2]

[1] Laboratoire d'informatique, UMR 7161 CNRS, Campus de l'École polytechnique 1, rue Honoré d'Estienne d'Orves Bâtiment Alan Turing, CS35003, 91120 Palaiseau, France
vdhoeven@lix.polytechnique.fr
[2] Inria Sophia Antipolis Méditerranée, AROMATH, 2004 route des Lucioles, 06902 Sophia Antipolis, France
bernard.mourrain@inria.fr

Abstract. In this paper, we present an efficient algorithm for the certification of numeric solutions to eigenproblems. The algorithm relies on a mixture of ball arithmetic, a suitable Newton iteration, and clustering of eigenvalues that are close.

Keywords: Ball arithmetic · Interval arithmetic · Reliable computing Computable analysis

A.M.S. Subject Classification: 65G20 · 03F60 · 65F99

1 Introduction

Let $\mathbb{F}$ be the set of floating point numbers for a fixed precision and a fixed exponent range. We will denote $\mathbb{F}^{\geqslant} = \{x \in \mathbb{F} : x \geqslant 0\}$. Consider an $n \times n$ matrix $M \in \mathbb{F}[\mathrm{i}]^{n \times n}$ with complex floating entries. The numeric *eigenproblem* associated to M is to compute a transformation matrix $T \in \mathbb{F}[\mathrm{i}]^{n \times n}$ and a diagonal matrix $D \in \mathbb{F}[\mathrm{i}]^{n \times n}$ such that

$$D \approx T^{-1} M T. \tag{1}$$

The entries of D are the approximate eigenvalues and the columns of T are the approximate eigenvectors of M. In addition, we might require that T is normalized. For instance, each of the columns might have unit norm. Alternatively, the norm of the i-th column may be required to be the same as the norm of the i-th row of T^{-1}, for each i. There are several well-known algorithms for solving the numeric eigenproblem [6].

Unfortunately, (1) is only an approximate equality. It is sometimes important to have rigourous bounds for the distance between the approximate eigenvalues

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 81–94, 2017.
https://doi.org/10.1007/978-3-319-72453-9_6

and/or eigenvectors and the genuine ones. More precisely, we may ask for a diagonal matrix $D_r \in (\mathbb{F}^{\geqslant})^{n\times n}$ and a matrix $T_r \in (\mathbb{F}^{\geqslant})^{n\times n}$ such that there exists a matrix $T' \in \mathbb{C}^{n\times n}$ for which

$$D' = (T')^{-1} M T$$

is diagonal and

$$|D'_{i,i} - D_{i,i}| \leqslant (D_r)_{i,i}$$
$$|T'_{i,j} - T_{i,j}| \leqslant (T_r)_{i,j}$$

for all i, j. This task will be called the *certification problem* of the numeric solution (D, T) to the eigenproblem for M. The matrices D_r and T_r can be thought of as reliable error bounds for the numerical solution (D, T) of the eigenproblem.

It will be convenient to rely on *ball arithmetic* [11,14], which is a systematic technique for this kind of bound computations. When computing with complex numbers, ball arithmetic is more accurate than more classical interval arithmetic [1,13,15,17,18,21], especially in multiple precision contexts. We will write $\mathbb{B} = \mathcal{B}(\mathbb{F}[\mathrm{i}], \mathbb{F}^{\geqslant})$ for the set of balls $\boldsymbol{z} = \mathcal{B}(z_c, z_r) = \{z \in \mathbb{C} : |z - z_c| \leqslant z_r\}$ with centers z_c in $\mathbb{F}[\mathrm{i}]$ and radii z_r in $\mathbb{F}^{\geqslant}$. In a similar way, we may consider matricial balls $\boldsymbol{M} = \mathcal{B}(M_c, M_r) \in \mathcal{B}(\mathbb{F}[\mathrm{i}]^{n\times n}, (\mathbb{F}^{\geqslant})^{n\times n})$: given a center matrix $M_c \in \mathbb{F}[\mathrm{i}]^{n\times n}$ and a radius matrix $M_r \in (\mathbb{F}^{\geqslant})^{n\times n}$, we have

$$\boldsymbol{M} = \mathcal{B}(M_c, M_r) = \{M \in \mathbb{C}^{n\times n} : \forall i, j, |(M_c)_{i,j} - M_{i,j}| \leqslant (M_r)_{i,j}\}.$$

Alternatively, we may regard $\mathcal{B}(M_c, M_r)$ as the set of matrices in $\mathbb{B}^{n\times n}$ with ball coefficients:

$$\mathcal{B}(M_c, M_r)_{i,j} = \mathcal{B}((M_c)_{i,j}, (M_r)_{i,j}).$$

Standard arithmetic operations on balls are carried out in a reliable way. For instance, if $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{B}$, then the computation of the product $\boldsymbol{w} = \boldsymbol{u}\boldsymbol{v}$ using ball arithmetic has the property that $uv \in \boldsymbol{w}$ for any $u \in \boldsymbol{u}$ and $v \in \boldsymbol{v}$. Given a ball $\boldsymbol{z} \in \mathbb{B}$, it will finally be convenient to write $\lfloor \boldsymbol{z} \rfloor \in \mathbb{F}^{\geqslant}$ and $\lceil \boldsymbol{z} \rceil \in \mathbb{F}^{\geqslant}$ for certified lower and upper bounds of $|\boldsymbol{z}|$ in $\mathbb{F}^{\geqslant}$.

In the language of ball arithmetic, it is natural to allow for small errors in the input and replace the numeric input $M \in \mathbb{F}[\mathrm{i}]^{n\times n}$ by a ball input $\mathcal{B}(M_c, M_r) \in \mathbb{B}^{n\times n}$. Then we may still compute a numeric solution

$$D_c \approx T_c^{-1} M_c T_c, \tag{2}$$

for the eigenproblem associated to the center M_c. Assume that the matrices in $\mathcal{B}(M_c, M_r)$ are all diagonalizable. The generalized *certification problem* now consists of the computation of a diagonal matrix $D_r \in (\mathbb{F}^{\geqslant})^{n\times n}$ and a matrix $T_r \in \mathbb{F}[\mathrm{i}]^{n\times n}$ such that, for every $M \in \mathcal{B}(M_c, M_r)$, there exist $D \in \mathcal{B}(D_c, D_r)$ and $T \in \mathcal{B}(T_c, T_r)$ with

$$D = T^{-1} M T.$$

In absence of multiple eigenvalues, known algorithms for solving this problem such as [20,23] proceed by the individual certification of each eigenvector, which results in an $O(n^4)$ running time. From the more theoretical perspective of α-theory [3], we also refer to [2] for numerically stable, strongly accurate, and theoretically efficient algorithms for solving eigenproblems.

Extensions to a cluster of eigenvalues and the corresponding eigenvectors have been considered in [4,22], with similar $O(n^4)$ complexity bounds. Fixed points theorem based on interval arithmetic are used to prove the existence of a matrix with a given Jordan block in the matrix interval domain. Such an approach has been exploited for the analysis of multiple roots in [7,19]. A test that provides an enclosing of all the eigenvalues has been proposed in [16]. Its certification relies on interval and ball arithmetics. The complexity of the test is in $O(n^3)$ but no iteration converging to the solution of the eigenproblem is described.

In this paper, we present a new algorithm of time complexity $O(n^3)$ for certifying and enclosing clusters of eigenvectors and eigenvalues in a single step. We also provide an iterative procedure that converges geometrically to clusters of solutions. This convergence is quadratic in the case of single eigenvalues. Our algorithm extends a previous algorithm from [11] to the case of multiple eigenvalues. This yields an efficient test for *approximate eigenvalues*.

From a more theoretical bit complexity point of view, our algorithm essentially reduces the certification problem to a constant number of numeric matrix multiplications. When using a precision of p bits for numerical computations, it has recently been shown [9] that two $n \times n$ matrices can be multiplied in time $\mathsf{MM}(n,p) = O\left(n^2\mathsf{I}(p) + n^\omega p 2^{O(\lg^* p - \lg^* n)}\mathsf{I}(\lg n)/\lg n\right)$. Here $\mathsf{I}(p) = O(p \lg p K^{\lg^* p})$ with $K \leqslant 6$ is the cost of p-bit integer multiplication [8,10] and $\omega < 2.3728639$ is the exponent of matrix multiplication [5]. If p is large enough with respect to the log of the condition number, then $O\left(\mathsf{MM}(n,p)\right)$ yields an asymptotic bound for the bit complexity of our certification problem.

We recall that it is very unlikely that the numeric matrix $M_c \in \mathbb{F}[\mathrm{i}]^{n\times n}$ with complex floating point coefficients has multiple eigenvalues. Indeed, small perturbations of matrices with multiple eigenvalues, as induced by rounding errors, generically only have simple eigenvalues. Consequently, we may assume without loss of generality that the numeric eigenproblem (2) has a reasonably accurate solution (if necessary, we may slightly perturb M_c and increase M_r accordingly). Using ball arithmetic, it is straightforward to compute the matricial ball

$$\mathcal{B}(N_c, N_r) = \mathcal{B}(T_c, 0)^{-1}\mathcal{B}(M_c, M_r)\mathcal{B}(T_c, 0).$$

If our numerical algorithm is accurate, then the non diagonal entries of $\mathcal{B}(N_c, N_r)$ tend to be small, whence $\mathcal{B}(N_c, N_r)$ can be considered as a small perturbation of a diagonal matrix. If we can estimate how far eigenvalues and eigenvectors of diagonal matrices can drift away under small perturbations, we thus obtain a solution to the original certification problem.

Section 2 introduces notations. In Sect. 3, we perform a detailed study of the eigenproblem for small perturbations M of diagonal matrices. We exhibit

a Newton iteration for finding the solutions. This iteration has quadratic convergence in the absence of multiple eigenvalues and is also an efficient tool for doubling the precision of a solution. However, in the case of multiple eigenvalues, the eigenproblem is ill-posed. Indeed, by a well-known observation, *any* vector occurs as the eigenvector of a small perturbation of the 2×2 identity matrix. The best we can hope for is to group eigenvectors with close eigenvalues together in "clusters" (see also [22]) and only require $T^{-1}MT$ to be block diagonal. For this reason, we present our Newton iteration in a sufficiently general setting which encompasses block matrices. We will show that the iteration still admits geometric convergence for sufficiently small perturbations and that the blockwise certification is still sufficient for the computation of rigourous error bounds for the eigenvalues. In Sect. 4, we will present explicit algorithms for clustering and the overall certification problem.

In absence of multiple eigenvalues, the ANALYZIZ library of the MATHEMAGIX system [12] contains an efficient implementation of our algorithm. The new algorithm from this paper has still to be integrated.

2 Notations

2.1 Matrix Norms

Throughout this paper, we will use the max norm for vectors and the corresponding matrix norm. More precisely, given a vector $v \in \mathbb{C}^n$ and an $n \times n$ matrix $M \in \mathbb{C}^{n \times n}$, we set

$$\begin{aligned} \|v\| &= \max\{|v_1|, \ldots, |v_n|\} \\ \|M\| &= \max_{\|v\|=1} \|Mv\|. \end{aligned}$$

For a second matrix $N \in \mathbb{C}^{n \times n}$, we clearly have

$$\begin{aligned} \|M + N\| &\leqslant \|M\| + \|N\| \\ \|MN\| &\leqslant \|M\|\|N\|. \end{aligned}$$

Explicit machine computation of the matrix norm is easy using the formula

$$\|M\| = \max\{|M_{i,1}| + \cdots + |M_{i,n}| : 1 \leqslant i \leqslant n\}. \tag{3}$$

In particular, when changing certain entries of a matrix M to zero, its matrix norm $\|M\|$ can only decrease.

2.2 Clustering

Assume that we are given a partition

$$\{1, \ldots, n\} = I_1 \amalg \cdots \amalg I_p. \tag{4}$$

Such a partition will also be called a *clustering* and denoted by I. Two indices i, j are said to belong to the same *cluster* if there exists a k with $\{i, j\} \subseteq I_k$ and we will write $i \sim j$. Two entries $M_{i,j}$ and $M_{i',j'}$ of a matrix $M \in \mathbb{C}^{n\times n}$ are said to belong to the same *block* if $i \sim j$ and $i' \sim j'$. We thus regard M as a generalized block matrix, for which the rows and columns of the blocks are not necessarily contiguous inside M.

A matrix $M \in \mathbb{C}^{n\times n}$ is said to be *block diagonal* (relative to the clustering) if $M_{i,j} = 0$ whenever $i \nsim j$. Similarly, we say that M is *off block diagonal* if $M_{i,j} = 0$ whenever $i \sim j$. For a general $M \in \mathbb{C}^{n\times n}$, we define its block diagonal and off block diagonal projections $\Delta(M) = \Delta^I(M)$ and $\Omega(M) = \Omega^I(M)$ by

$$\Delta(M)_{i,j} = \begin{cases} M_{i,j} \text{ if } i \sim j \\ 0 \quad \text{otherwise} \end{cases} \qquad \Omega(M)_{i,j} = \begin{cases} 0 \quad \text{if } i \sim j \\ M_{i,j} \text{ otherwise} \end{cases}$$

By our observation at the end of Sect. 2.1, we have

$$\|\Delta(M)\| \leqslant \|M\|$$
$$\|\Omega(M)\| \leqslant \|M\|.$$

For the *trivial clustering* $I_k = \{k\}$, the matrices $\Delta(M)$ and $\Omega(M)$ are simply the diagonal and off diagonal projections of M. In that case we will also write $\Delta^* = \Delta$ and $\Omega^* = \Omega$.

2.3 Diagonal Matrices

Below, we will study eigenproblems for perturbations of a given diagonal matrix

$$D = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}. \tag{5}$$

It follows from (3) that the matrix norm $\mu = \|D\|$ of a diagonal matrix D is given by

$$\mu = \max\{|\lambda_1|, \ldots, |\lambda_n|\}.$$

It will also be useful to define the *separation number* $\sigma^* = \sigma^*(D)$ by

$$\sigma^* = \min\{|\lambda_i - \lambda_j| : i \neq j\}.$$

More generally, given a clustering as in the previous subsection, we also define the *block separation number* $\sigma = \sigma(D) = \sigma^I(D)$ by

$$\sigma = \min\{|\lambda_i - \lambda_j| : i \nsim j\}$$

This number σ remains high if the clustering is chosen in such a way that the indices i, j of any two "close" eigenvalues λ_i and λ_j belong to the same cluster. In particular, if $\sigma > 0$, then $\lambda_i = \lambda_j$ implies $i \sim j$.

3 Eigenproblems for Perturbed Diagonal Matrices

3.1 The Linearized Equation

Let D be a diagonal matrix (5). Given a small perturbation

$$M = D + H$$

of D, where H is an off diagonal matrix, the aim of this section is to find a small matrix $E \in \mathbb{C}^{n\times n}$ for which

$$M' = (1+E)^{-1}M(1+E)$$

is block diagonal. In other words, we need to solve the equation

$$\Omega((1+E)^{-1}(D+H)(1+E)) = 0.$$

When linearizing this equation in E and H, we obtain

$$\Omega([D,E]+H) = 0.$$

If E is strongly off diagonal, then so is $[D,E]$, and the equation further reduces to

$$[D,E] = -\Omega(H).$$

This equation can be solved using the following lemma:

Lemma 1. *Given a matrix $A \in \mathbb{C}^{n\times n}$ and a diagonal matrix D with entries $\lambda_1, \ldots, \lambda_n$, let $B = \Phi(D,A) \in \mathbb{C}^{n\times n}$ be the strongly off diagonal matrix with*

$$B_{i,j} = \begin{cases} 0 & \text{if } i \sim j \\ \frac{A_{i,j}}{\lambda_j - \lambda_i} & \text{otherwise} \end{cases}$$

Then $\|B\| \leqslant \sigma^{-1}\|A\|$ and

$$[D,B] = -\Omega(A). \tag{6}$$

Proof. The inequality follows from (3) and the definition of σ. One may check (6) using a straightforward computation.

3.2 The Fundamental Iteration

In view of the lemma, we now consider the iteration

$$(D,H) \longmapsto (D',H'),$$

where

$$\begin{aligned} E &= \Phi(D,H) \\ M' &= (1+E)^{-1}(D+H)(1+E) \\ D' &= \Delta^*(M') \\ H' &= \Omega^*(M') \end{aligned}$$

In order to study the convergence of this iteration, we introduce the quantities

$$\begin{array}{ll} \mu = \|D\| & \mu' = \|D'\| \\ \sigma = \sigma(D) & \sigma' = \sigma(D') \\ \eta_1 = \|\Delta(H)\| & \eta_1' = \|\Delta(H')\| \\ \eta_2 = \|\Omega(H)\| & \eta_2' = \|\Omega(H')\| \end{array}$$

$$\alpha = \min\left\{\tfrac{\sigma}{6\mu}, \tfrac{1}{4}\right\}.$$

Lemma 2. *For $\delta \in (0,1]$, assume that*

$$\begin{aligned} \eta_1 + \eta_2 &\leqslant \alpha\delta\mu \\ \eta_2 &\leqslant \alpha\delta\sigma. \end{aligned}$$

Then $\|D' - D\| \leqslant \delta\eta_2$ and

$$\begin{aligned} \mu' &\leqslant \mu + \delta\eta_2 \\ \sigma' &\geqslant \sigma - 2\delta\eta_2 \\ \eta_1' &\leqslant \eta_1 + \delta\eta_2 \\ \eta_2' &\leqslant \delta\eta_2. \end{aligned}$$

Proof. We have

$$\begin{aligned} M' - D &= H + [D,E] + R \\ &= \Delta(H) + R, \end{aligned}$$

where

$$R = E^2(1+E)^{-1}(D+H)(1+E) - E(D+H)E + [H,E].$$

Setting $\varepsilon = \|E\| \leqslant \sigma^{-1}\eta_2 \leqslant \alpha\delta \leqslant \frac{1}{4}$, the remainder R is bounded by

$$\begin{aligned} \|R\| &\leqslant \varepsilon^2 \frac{1}{1-\varepsilon}(1+\alpha\delta)\mu(1+\varepsilon) + \varepsilon(1+\alpha\delta)\mu\varepsilon + 2(\eta_1+\eta_2)\varepsilon \\ &= \frac{2\varepsilon^2}{1-\varepsilon}(1+\alpha\delta)\mu + 2(\eta_1+\eta_2)\varepsilon \\ &\leqslant (4\varepsilon\mu + 2\alpha\delta\mu)\varepsilon \\ &\leqslant 6\alpha\delta\mu\sigma^{-1}\eta_2 \\ &\leqslant \delta\eta_2. \end{aligned}$$

Consequently,

$$\begin{aligned} \|D' - D\| &= \|\Delta^*(M' - D)\| = \|\Delta^*(R)\| \\ &\leqslant \|R\| \leqslant \delta\eta_2 \\ \eta_1' &= \|\Delta(H')\| = \|\Omega^*(\Delta(M'))\| = \|\Omega^*(\Delta(H+R))\| \\ &\leqslant \|H+R\| \leqslant \eta_1 + \delta\eta_2 \\ \eta_2' &= \|\Omega(H')\| = \|\Omega(M')\| = \|\Omega(R)\| \\ &\leqslant \delta\eta_2. \end{aligned}$$

The inequalities $\mu' \leqslant \mu + \delta\eta_2$ and $\sigma' \geqslant \sigma - 2\delta\eta_2$ follow from $\|D' - D\| \leqslant \delta\eta_2$.

3.3 Convergence of the Fundamental Iteration

Theorem 1. *Assume that*

$$\begin{aligned} \eta_1 + \eta_2 &\leqslant \tfrac{1}{8}\alpha\mu \\ \eta_2 &\leqslant \tfrac{1}{8}\alpha\sigma. \end{aligned}$$

Then the sequence

$$(D, H), (D', H'), (D'', H''), \ldots$$

converges geometrically to a limit $(M^{(\infty)}, H^{(\infty)})$ *with* $\|D^{(\infty)} - M\| \leqslant \eta_2$ *and* $\|H^{(\infty)}\| \leqslant \eta_1 + \eta_2$. *The matrix* $D^{(\infty)} + H^{(\infty)}$ *is block diagonal and there exists a matrix* $\hat{E}$ *with* $\|\hat{E}\| \leqslant 3\sigma^{-1}\eta_2$, *such that*

$$D^{(\infty)} + H^{(\infty)} = (1 + \hat{E})^{-1}(D + H)(1 + \hat{E}).$$

Proof. Let $(D^{(i)}, H^{(i)})$ stand for the i-th fundamental iterate of (D, H) and $E^{(i)} = \Phi(H^{(i)}, D^{(i)})$. Denote $\mu^{(i)} = \|D^{(i)}\|$, $\sigma^{(i)} = \sigma(D^{(i)})$, $\eta_1^{(i)} = \|\Delta(H^{(i)})\|$ and $\eta_2^{(i)} = \|\Omega(H^{(i)})\|$. Let us show by induction over i that

$$\begin{aligned} \|D^{(i)} - D\| &\leqslant (1 - \tfrac{1}{2^i})\eta_2 \\ \mu^{(i)} &\leqslant \mu + (1 - \tfrac{1}{2^i})\eta_2 \\ \sigma^{(i)} &\geqslant \tfrac{1}{2}(1 + \tfrac{1}{2^i})\sigma \\ \eta_1^{(i)} &\leqslant \eta_1 + (1 - \tfrac{1}{2^i})\eta_2 \\ \eta_2^{(i)} &\leqslant \tfrac{1}{2^i}\eta_2. \end{aligned}$$

This is clear for $i = 0$. Assume that the induction hypothesis holds for a given i and let

$$\alpha^{(i)} = \min\left\{\frac{\sigma^{(i)}}{6\mu^{(i)}}, \frac{1}{4}\right\}$$

Since $(1 - \frac{1}{2^i})\eta_2 \leqslant \frac{1}{32}\mu$, the induction hypothesis implies

$$\begin{aligned} \mu^{(i)} &\leqslant 2\mu \\ \sigma^{(i)} &\geqslant \tfrac{1}{2}\sigma \\ \alpha^{(i)} &\geqslant \tfrac{1}{4}\alpha. \end{aligned}$$

Applying Lemma 2 for $(D^{(i)}, H^{(i)})$ and $\delta = \frac{1}{2}$, we thus find

$$\begin{aligned}
\|D^{(i+1)} - D\| &\leqslant \|D^{(i)} - D\| + \|D^{(i+1)} - D^{(i)}\| \\
&\leqslant (1 - \tfrac{1}{2^i})\eta_2 + \tfrac{1}{2^{i+1}}\eta_2 \leqslant (1 - \tfrac{1}{2^{i+1}})\eta_2 \\
\mu^{(i+1)} &\leqslant \mu^{(i)} + \tfrac{1}{2}\eta_2^{(i)} \leqslant \mu + (1 - \tfrac{1}{2^{i+1}})\eta_2 \\
\sigma^{(i+1)} &\geqslant \sigma^{(i)} - \tfrac{1}{2}\eta_2^{(i)} \geqslant \tfrac{1}{2}(1 + \tfrac{1}{2^i} - \tfrac{1}{2^{i+1}})\sigma \geqslant \tfrac{1}{2}(1 + \tfrac{1}{2^{i+1}})\sigma \\
\eta_1^{(i+1)} &\leqslant \eta_1^{(i)} + \tfrac{1}{2}\eta_2^{(i)} \leqslant \eta_1 + (1 - \tfrac{1}{2^{i+1}})\eta_2 \\
\eta_2^{(i+1)} &\leqslant \tfrac{1}{2}\eta_2^{(i)} \leqslant \tfrac{1}{2^{i+1}}\eta_2.
\end{aligned}$$

This completes the induction.

Applying the induction to the sequence starting at $D^{(i)}$, we have for every $j \geqslant 0$,

$$\|D^{(i+j)} - D^{(i)}\| \leqslant (1 - \tfrac{1}{2^{j+1}})\eta_2^{(i)} \leqslant (1 - \tfrac{1}{2^{j+1}})\tfrac{1}{2^{i+1}}\eta_2.$$

This shows that $D^{(i)}$ is a Cauchy sequence that tends to a limit $D^{(\infty)}$ with $\|D^{(\infty)} - D\| \leqslant \eta_2$. From this inequality, we also deduce that $\|D^{(\infty)} - D^{(i)}\| \leqslant \frac{1}{2^{i+1}}\eta_2$, so $D^{(i)}$ converges geometrically to $D^{(\infty)}$.

Moreover, for each i, we have $\varepsilon^{(i)} = \|E^{(i)}\| \leqslant \sigma^{-1}\eta_2^{(i)} \leqslant \frac{1}{2^i}\sigma^{-1}\eta_2$. Hence, the matrix

$$\hat{E} = (1 + E^{(0)})(1 + E^{(1)})(1 + E^{(2)}) \cdots - 1$$

is well defined, and

$$\begin{aligned}
\log(1 + \|\hat{E}\|) &\leqslant \log(1 + \varepsilon^{(0)}) + \log(1 + \varepsilon^{(1)}) + \log(1 + \varepsilon^{(2)}) + \cdots \\
&\leqslant 2\sigma^{-1}\eta_2.
\end{aligned}$$

We deduce that

$$\|\hat{E}\| \leqslant \mathrm{e}^{2\sigma^{-1}\eta_2} - 1 \leqslant 3\sigma^{-1}\eta_2,$$

since $\sigma^{-1}\eta_2 \leqslant \frac{1}{32}$.

We claim that $M^{(i)} = D^{(i)} + H^{(i)}$ converges geometrically to

$$M^{(\infty)} = (1 + \hat{E})^{-1} M^{(0)} (1 + \hat{E}).$$

For any matrix $M, E \in \mathbb{C}^{n\times n}$ with $\|E\| < \varepsilon < 1$, we have

$$\begin{aligned}
\|(1 + E)^{-1}M(1 + E) - M\| &= \|ME - E(1 + E)^{-1}M(1 + E)\| \\
&\leqslant \|M\|(\varepsilon + \varepsilon(1 + \varepsilon)\|(1 + E)^{-1}\|) \\
&\leqslant \varepsilon\|M\|(1 + (1 + \varepsilon)(1 - \varepsilon)^{-1}) \\
&= \frac{2\varepsilon}{1 - \varepsilon}\|M\|. \qquad (7)
\end{aligned}$$

Let $\hat{E}^{(i)} = (1 + E^{(i)})(1 + E^{(i+1)})(1 + E^{(i+2)}) \cdots - 1$. By the same arguments as above, we have $\hat{\varepsilon}_i := \|E^{(i)}\| \leqslant 3\sigma^{-1}\eta_2^{(i)} = \frac{3}{2^{i+1}}\sigma^{-1}\eta_2$. Since $M^{(\infty)} = (1 + \hat{E}^{(i)})^{-1}M^{(i)}(1 + \hat{E}^{(i)})$, the inequality (7) implies

$$\begin{aligned}\|M^{(\infty)} - M^{(i)}\| &\leqslant \frac{2\hat{\varepsilon}_i}{1 - \hat{\varepsilon}_i}(\|D^{(i)}\| + \|H^{(i)}\|)\\ &\leqslant \frac{2\hat{\varepsilon}_i}{1 - \hat{\varepsilon}_i}(\mu_i + \eta_1^{(i)} + \eta_2^{(i)})\\ &\leqslant \frac{3}{2^i}\frac{\sigma^{-1}\eta_2}{1 - \hat{\varepsilon}_i}(\mu + \eta_1 + \eta_2).\end{aligned}$$

This shows that $M^{(i)}$ converges geometrically to $M^{(\infty)}$. We deduce that the sequence $H^{(i)} = M^{(i)} - D^{(i)}$ also converges geometrically to a limit $H^{(\infty)}$ with $\|H^{(\infty)}\| \leqslant \eta_1 + \eta_2$. Since $\lim_{i\to\infty} \eta_2^{(i)} = 0$, we finally observe that $M^{(\infty)} = D^{(\infty)} + H^{(\infty)}$ is block diagonal.

Theorem 2. *Assume $I_k = \{k\}$ for all k. Then, under the assumptions of Theorem 1, the sequence $(D, H), (D', H'), (D'', H''), \ldots$ converges quadratically to $(D^{(\infty)}, 0)$.*

Proof. The extra assumption implies that $\eta_1^{(i)} = 0$ for all i. Let us show by induction over i that we now have

$$\eta_2^{(i)} \leqslant \tfrac{1}{2^{2^i-1}}\eta_2.$$

This is clear for $i = 0$. Assume that the result holds for a given i. Then we may apply Lemma 2 to $(D^{(i)}, H^{(i)})$ for $\delta = 2^{-2^i+1}$, and obtain

$$\begin{aligned}\eta_2^{(i+1)} &\leqslant \tfrac{1}{2^{2^i-1}}\eta_2^{(i)}\\ &\leqslant \tfrac{1}{2^{2^{i+1}-1}}.\end{aligned}$$

Since $\|D^{(i+1)} - D^{(i)}\| \leqslant \eta_2^{(i)}$, this establishes the quadratic convergence.

4 Algorithms

4.1 Clustering

Let $M = D + H$ be the perturbation of a diagonal matrix (5) as in the previous section. In order to apply Theorem 1, we first have to find a suitable clustering (4). For a given threshold separation δ, we will simply take the finest clustering (i.e. for which p is maximal) with the property that $|\lambda_i - \lambda_j| \leqslant \delta \Rightarrow i \sim j$. This clustering can be computed using the algorithm Cluster below.

Algorithm Cluster
Input: eigenvalues $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n \in \mathbb{B}$ and $\delta \in \mathbb{F}^{\geqslant}$
Output: the finest clustering (4) with $\lfloor \boldsymbol{\lambda}_i - \boldsymbol{\lambda}_j \rfloor \leqslant \delta \Rightarrow i \sim j$

- Let G be the graph with vertices $1, \ldots, n$ and such that i and j are connected if and only if $\lfloor \boldsymbol{\lambda}_i - \boldsymbol{\lambda}_j \rfloor \leqslant \delta$.
- Let H be the transitive closure of G.
- Let $H_1, \ldots, H_p$ the connected components of H.
- Let I_k be the set of vertices of H_k for each k.

4.2 Certification in the Case of Perturbed Diagonal Matrices

In order to apply Theorem 1, it now remains to find a suitable threshold δ for which the conditions of the theorem hold. Starting with $\delta = 0$, we will simply increase δ to $\sigma(D)$ whenever the conditions are not satisfied. This will force the number p of clusters to decrease by at least one at every iteration, whence the algorithm terminates. Notice that the workload of one iteration is $O(n^2)$, so the total running time remains bounded by $O(n^3)$.

Algorithm DiagonalCertify
Input: a diagonal ball matrix $\boldsymbol{D} \in \mathbb{B}^{n \times n}$ with entries $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n$ and an off diagonal ball matrix $\boldsymbol{H} \in \mathbb{B}^{n \times n}$
Output: a clustering I and $\hat{\varepsilon} \in \mathbb{F}$ such that, for any $M \in \boldsymbol{D}$ and $H \in \boldsymbol{H}$, the conditions of theorem 1 hold and $\|\hat{E}\| \leqslant \hat{\varepsilon}$

$\delta := 0$
Repeat
  Compute the clustering I for $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n$ and δ using Cluster
  Let $\boldsymbol{\mu} := \|\boldsymbol{D}\|$, $\boldsymbol{\sigma} := \sigma^I(\boldsymbol{D})$, $\boldsymbol{\eta}_1 := \|\Delta^I(\boldsymbol{H})\|$ and $\boldsymbol{\eta}_2 := \|\Omega^I(\boldsymbol{H})\|$
  Let $\boldsymbol{\alpha} := \min\left\{\frac{\boldsymbol{\sigma}}{6\boldsymbol{\mu}}, \frac{1}{4}\right\}$
  If $\lceil \boldsymbol{\eta}_1 + \boldsymbol{\eta}_2 \rceil \leqslant \lfloor \frac{\boldsymbol{\alpha}\boldsymbol{\mu}}{8} \rfloor$ and $\lceil \boldsymbol{\eta}_2 \rceil \leqslant \lfloor \frac{\boldsymbol{\alpha}\boldsymbol{\sigma}}{8} \rfloor$, then return $(I, \lceil \frac{3\boldsymbol{\eta}_2}{\boldsymbol{\sigma}} \rceil)$
  Set $\delta := \lceil \boldsymbol{\sigma} \rceil$

4.3 Certification of Approximate Eigenvectors and Eigenvalues

Let us now return to the original problem of certifying a numerical solution to an eigenproblem. We will denote by $\mathbb{1}_n$ the $n \times n$ matrix of which all entries are one.

Algorithm EigenvectorCertify
Input: $\boldsymbol{M} = \mathcal{B}(M_c, M_r) \in \mathbb{B}^{n\times n}$ and $T_c \in \mathbb{F}[\mathrm{i}]^{n\times n}$ such that $T_c^{-1} M_c T_c$ is approximately diagonal
Output: a clustering I and $\boldsymbol{T} = \mathcal{B}(T_c, T_r) \in \mathbb{B}^{n\times n}$ such that for any $M \in \boldsymbol{M}$, there exists a $T \in \boldsymbol{T}$ for which $T^{-1}MT$ is block diagonal

Compute $\boldsymbol{D} := \mathcal{B}(T_c, 0)^{-1} \boldsymbol{M} \mathcal{B}(T_c, T_r)$
Let $(I, \varepsilon) :=$ DiagonalCertify$(\Delta^*(\boldsymbol{D}), \Omega^*(\boldsymbol{D}))$
Let $\boldsymbol{E} := \mathcal{B}(T_c, 0)\mathcal{B}(0, \varepsilon)\mathbb{1}_n$
Let $(T_r)_{i,j} := \lceil \boldsymbol{E}_{i,j} \rceil$ for all i, j
Return $(I, \mathcal{B}(T_c, T_r))$

Obviously, any eigenvalue $\lambda \in \mathbb{C}$ of a matrix $M \in \mathbb{C}^{n\times n}$ satisfies $|\lambda| \leqslant \|M\|$. We may thus use the following modification of EigenvectorCertify in order to compute enclosures for the eigenvalues of $\boldsymbol{M}$.

Algorithm EigenvalueCertify
Input: $\boldsymbol{M} = \mathcal{B}(M_c, M_r) \in \mathbb{B}^{n\times n}$ and $T_c \in \mathbb{F}[\mathrm{i}]^{n\times n}$ such that $T_c^{-1} M_c T_c$ is approximately diagonal
Output: ball enclosures $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n \in \mathbb{B}$ for the eigenvalues of $\boldsymbol{M}$, with the appropriate multiplicities in cases of overlapping

Compute $\boldsymbol{D} := \mathcal{B}(T_c, 0)^{-1} \boldsymbol{M} \mathcal{B}(T_c, T_r)$
Let $(I, \varepsilon) :=$ DiagonalCertify$(\Delta^*(\boldsymbol{D}), \Omega^*(\boldsymbol{D}))$
Let $\boldsymbol{\eta}_1 := \|\Delta^I(\Omega^*(\boldsymbol{D}))\|$ and $\boldsymbol{\eta}_2 := \|\Omega^I(\Omega^*(\boldsymbol{D}))\|$
For each $k \in \{1, \ldots, p\}$ do
  If $I_k = \{i\}$ for some i, then let $\boldsymbol{\lambda}_i := \mathcal{B}((D_c)_{i,i}, \lceil \boldsymbol{\eta}_2 \rceil)$
  Otherwise
    Let $\boldsymbol{c}$ be the barycenter of the $\boldsymbol{D}_{i,i}$ with $i \in I_k$
    Let $\boldsymbol{r}$ be the maximum of $|\boldsymbol{D}_{i,i} - c|$ for $i \in I_k$
    Let $\boldsymbol{\lambda}_i := \boldsymbol{c} + \mathcal{B}(0, \lceil \boldsymbol{r} + \boldsymbol{\eta}_1 + 2\boldsymbol{\eta}_2 \rceil)$ for all $i \in I_k$
Return $(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n)$

5 Possible Extensions

Let $M \in \mathbb{C}^{n\times n}$ be a matrix with a (numerically) multiple eigenvalue λ. We have already stressed that it is generally impossible to provide non trivial certifications for the corresponding eigenvectors. Nevertheless, two observations should be made:

- If the eigenspace E_λ corresponding to λ has dimension 1, then small perturbations of the matrix M only induce small perturbations of λ and E_λ.
- Let F_λ denote the full invariant subspace associated to the eigenvalue λ (or all eigenvalues in the cluster of λ). Then small perturbations of M only induce small perturbations of λ and F_λ.

More precisely, in these two cases, we may search for ball enclosures for orthonormal bases of the vector spaces E_λ resp. F_λ, which do not contain the zero vector.

When considering the numeric solution (1) of the eigenproblem for M, the column vectors which generate F_λ are usually far from being orthogonal. Orthonormalization can only be done at the expense of making $T^{-1}MT$ only upper triangular. Moreover, the orthogonalization implies a big loss of accuracy, which requires the application of a correction method for restoring the accuracy. It seems that the fundamental Newton iteration from Sect. 3.2 can actually be used as a correction method. For instance, for small perturbations of the matrix

$$D = \begin{pmatrix} \lambda_1 & 1 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & \lambda_2 & 1 \\ 0 & 0 & 0 & \lambda_2 \end{pmatrix},$$

it can be shown that the fundamental iteration still converges. However, for more general block diagonal matrices with triangular blocks, the details are quite technical and yet to be worked out.

Yet another direction for future investigations concerns the quadratic convergence. As a refinement of Lemma 1, we might replace D by a block diagonal matrix with entries $\Lambda_1, \ldots, \Lambda_p$. Instead of taking $B_{i,j} = \frac{M_{i,j}}{\lambda_j - \lambda_i}$, we then have to solve equations of the form

$$B_{i,j}\Lambda_j - \Lambda_i B_{i,j} = M_{i,j}.$$

If the Λ_i are sufficiently close to $\lambda_i \,\mathrm{Id}$, it might then be possible to adapt the fundamental iteration accordingly so as to achieve quadratic convergence for the strongly off diagonal part.

References

1. Alefeld, G., Herzberger, J.: Introduction to Interval Analysis. Academic Press, New York (1983)
2. Armentano, D., Beltrán, C., Bürgisser, P., Cucker, F., Shub, M.: A stable, polynomial-time algorithm for the eigenpair problem. Technical report, arXiv (2014). http://arxiv.org/abs/1505.03290
3. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and Real Computation. Springer, New York (1998). https://doi.org/10.1007/978-1-4612-0701-6
4. Dongarra, J.J., Moler, C.B., Wilkinson, J.H.: Improving the accuracy of computed eigenvalues and eigenvectors. SIAM J. Numer. Anal. **20**(1), 23–45 (1983)
5. Le Gall, F.: Powers of tensors and fast matrix multiplication. In: Proceedings of ISSAC 2014, Kobe, Japan, pp. 296–303, 23–25 July 2014
6. Golub, G.H., Van Loan, F.: Matrix Computations. JHU Press, Baltimore (1996)
7. Graillat, S., Trébuchet, P.: A new algorithm for computing certified numerical approximations of the roots of a zero-dimensional system. In: Proceedings of ISSAC 2009, pp. 167–174. ACM Press (2009)
8. Harvey, D.: Faster truncated integer multiplication (2017). https://arxiv.org/abs/1703.00640

9. Harvey, D., van der Hoeven, J.: On the complexity of integer matrix multiplication. Technical report, HAL (2014, accepted for publication in JSC). http://hal.archives-ouvertes.fr/hal-01071191
10. Harvey, D., van der Hoeven, J., Lecerf, G.: Even faster integer multiplication. J. Complex. **36**, 1–30 (2016)
11. van der Hoeven, J.: Ball arithmetic. Technical report, HAL (2009). http://hal.archives-ouvertes.fr/hal-00432152
12. van der Hoeven, J., Lecerf, G., Mourrain, B., et al.: Mathemagix (2002). http://www.mathemagix.org
13. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis. Springer, London (2001). https://doi.org/10.1007/978-1-4471-0249-6
14. Johansson, F.: Arb: a C library for ball arithmetic. ACM Commun. Comput. Algebra **47**(3/4), 166–169 (2014)
15. Kulisch, U.W.: Computer Arithmetic and Validity: Theory, Implementation, and Applications. Studies in Mathematics, vol. 33. de Gruyter, Berlin (2008)
16. Miyajima, S.: Fast enclosure for all eigenvalues in generalized eigenvalue problems. J. Comput. Appl. Math. **233**(11), 2994–3004 (2010)
17. Moore, R.E.: Interval Analysis. Prentice Hall, Englewood Cliffs (1966)
18. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge University Press, Cambridge (1990)
19. Rump, S., Graillat, S.: Verified error bounds for multiple roots of systems of nonlinear equations. Numer. Algorithms **54**, 359–377 (2010)
20. Rump, S.M.: Guaranteed inclusions for the complex generalized eigenproblem. Computing **42**(2), 225–238 (1989)
21. Rump, S.M.: INTLAB - INTerval LABoratory. In: Csendes, T. (ed.) Developments in Reliable Computing, pp. 77–104. Kluwer Academic Publishers, Dordrecht (1999). http://www.ti3.tu-harburg.de/rump/
22. Rump, S.M.: Computational error bounds for multiple or nearly multiple eigenvalues. Linear Algebra Appl. **324**(1–3), 209–226 (2001)
23. Yamamoto, T.: Error bounds for computed eigenvalues and eigenvectors. Numerische Mathematik **34**(2), 189–199 (1980)

Fast Chinese Remaindering in Practice

Joris van der Hoeven(✉)

Laboratoire d'informatique, UMR 7161 CNRS, Campus de l'École polytechnique,
1, rue Honoré d'Estienne d'Orves Bâtiment Alan Turing, CS35003,
91120 Palaiseau, France
vdhoeven@lix.polytechnique.fr

Abstract. The Chinese remainder theorem is a key tool for the design of efficient multi-modular algorithms. In this paper, we study the case when the moduli $m_1, \ldots, m_\ell$ are fixed and can even be chosen by the user. If ℓ is small or moderately large, then we show how to choose *gentle moduli* that allow for speedier Chinese remaindering. The multiplication of integer matrices is one typical application where we expect practical gains for various common matrix dimensions and bitsizes of the coefficients.

Keywords: Chinese remainder theorem · Algorithm · Complexity
Integer matrix multiplication

1 Introduction

Modular reduction is an important tool for speeding up computations in computer arithmetic, symbolic computation, and elsewhere. The technique allows to reduce a problem that involves large integer or polynomial coefficients to one or more similar problems that only involve small modular coefficients. Depending on the application, the solution to the initial problem is reconstructed *via* the Chinese remainder theorem or Hensel's lemma. We refer to [9, Chap. 5] for a gentle introduction to this topic.

In this paper, we will mainly be concerned with multi-modular algorithms over the integers that rely on the Chinese remainder theorem. Given $a, m \in \mathbb{Z}$ with $m > 1$, we will denote by $a \operatorname{rem} m \in \mathcal{R}_m := \{0, \ldots, m-1\}$ the remainder of the Euclidean division of a by m. Given an $r \times r$ matrix $A \in \mathbb{Z}^{r \times r}$ with integer coefficients, we will also denote $A \operatorname{rem} m \in \mathbb{Z}^{r \times r}$ for the matrix with coefficients $(A \operatorname{rem} m)_{i,j} = A_{i,j} \operatorname{rem} m$.

One typical application of Chinese remaindering is the multiplication of $r \times r$ integer matrices $A, B \in \mathbb{Z}^{r \times r}$. Assuming that we have a bound M with $2|(AB)_{i,j}| < M$ for all i, j, we proceed as follows:

1. Select moduli $m_1, \ldots, m_\ell$ with $m_1 \cdots m_\ell > M$ that are mutually coprime.
2. Compute $A \operatorname{rem} m_k$ and $B \operatorname{rem} m_k$ for $k = 1, \ldots, \ell$.
3. Multiply $C \operatorname{rem} m_k := (A \operatorname{rem} m_k)(B \operatorname{rem} m_k) \operatorname{rem} m_k$ for $k = 1, \ldots, \ell$.
4. Reconstruct $C \operatorname{rem} M$ from the $C \operatorname{rem} m_k$ with $k = 1, \ldots, \ell$.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 95–106, 2017.
https://doi.org/10.1007/978-3-319-72453-9_7

The simultaneous computation of $A_{i,j}$ rem m_k from $A_{i,j}$ for all $k = 1, \ldots, \ell$ is called the problem of *multi-modular reduction*. In step 1, we need to perform $2r^2$ multi-modular reductions for the coefficients of A and B. The inverse problem of reconstructing $C_{i,j}$ rem M from the $C_{i,j}$ rem m_k with $k = 1, \ldots, \ell$ is called the problem of *multi-modular reconstruction*. We need to perform r^2 such reconstructions in step 3. Our hypothesis on M allows us to recover C from C rem M.

Let us quickly examine when and why the above strategy pays off. In this paper, the number ℓ should be small or moderately large, say $\ell \leqslant 64$. The moduli $m_1, \ldots, m_\ell$ typically fit into a machine word. Denoting by μ the bitsize of a machine word (say $\mu = 32$ or $\mu = 64$), the coefficients of A and B should therefore be of bitsize $\approx \ell\mu/2$.

For small ℓ, integer multiplications of bitsize $\mu\ell/2$ are usually carried out using a naive algorithm, of complexity $\Theta(\ell^2)$. If we directly compute the product AB using r^3 naive integer multiplications, the computation time is therefore of order $\Theta(r^3\ell^2)$. In comparison, as we will see, one naive multi-modular reduction or reconstruction for ℓ moduli roughly requires $\Theta(\ell^2)$ machine operations, whereas an $r \times r$ matrix multiplication modulo any of the m_k can be done in time $\Theta(r^3)$. Altogether, this means that the above multi-modular algorithm for integer matrix multiplication has running time $\Theta(\ell^2 r^2 + r^3\ell)$, which is $\Theta(\min(\ell, r))$ times faster than the naive algorithm.

If $\ell \ll r$, then the cost $\Theta(\ell^2 r^2)$ of steps 1 and 3 is negligible with respect to the cost $\Theta(r^3\ell)$ of step 2. However, if ℓ and r are of the same order of magnitude, then Chinese remaindering may take an important part of the computation time; the main purpose of this paper is to reduce this cost. If $\ell \gg r$, then we notice that other algorithms for matrix multiplication usually become faster, such as naive multiplication for small ℓ, Karatsuba multiplication [13] for larger ℓ, or FFT-based techniques [6] for very large ℓ.

Two observations are crucial for reducing the cost of Chinese remaindering. First of all, the moduli $m_1, \ldots, m_\ell$ are the same for all $2r^2$ multi-modular reductions and r^2 multi-modular reconstructions in steps 1 and 3. If r is large, then this means that we can essentially assume that $m_1, \ldots, m_\ell$ were fixed once and for all. Secondly, we are free to choose $m_1, \ldots, m_\ell$ in any way that suits us. We will exploit these observations by precomputing *gentle moduli* for which Chinese remaindering can be performed more efficiently than for ordinary moduli.

The first idea behind gentle moduli is to consider moduli m_i of the form $2^{sw} - \varepsilon_i^2$, where w is somewhat smaller than μ, where s is even, and $\varepsilon_i^2 < 2^w$. In Sect. 3.1, we will show that multi-modular reduction and reconstruction both become a lot simpler for such moduli. Secondly, each m_i can be factored as $m_i = (2^{sw/2} - \varepsilon_i)(2^{sw/2} + \varepsilon_i)$ and, if we are lucky, then both $2^{sw/2} - \varepsilon_i$ and $2^{sw/2} + \varepsilon_i$ can be factored into $s/2$ moduli of bitsize $< \mu$. If we are very lucky, then this allows us to obtain $w\ell$ moduli $m_{i,j}$ of bitsize $\approx w$ that are mutually coprime and for which Chinese remaindering can be implemented efficiently.

Let us briefly outline the structure of this paper. In Sect. 2, we rapidly recall basic facts about Chinese remaindering and naive algorithms for this task. In Sect. 3, we introduce gentle moduli and describe how to speed up Chinese

remaindering with respect to such moduli. The last Sect. 4 is dedicated to the brute force search of gentle moduli for specific values of s and w. We implemented a sieving method in MATHEMAGIX which allowed us to compute tables with gentle moduli. For practical purposes, it turns out that gentle moduli exist in sufficient number for $s \leqslant 8$. We expect our technique to be efficient for $\ell \lesssim s^2$, but this still needs to be confirmed *via* an actual implementation. The application to integer matrix multiplication in Sect. 4.3 also has not been implemented yet.

Let us finally discuss a few related results. In this paper, we have chosen to highlight integer matrix multiplication as one typical application in computer algebra. Multi-modular methods are used in many other areas and the operations of multi-modular reduction and reconstruction are also known as conversions between the positional number system (PNS) and the residue number system (RNS). Asymptotically fast algorithms are based on *remainder trees* [3,8,14], with recent improvements in [2,4,10]; we expect such algorithms to become more efficient when ℓ exceeds s^2.

Special moduli of the form $2^n - \varepsilon$ are also known as *pseudo-Mersenne moduli*. They have been exploited before in cryptography [1] in a similar way as in Sect. 3.1, but with a slightly different focus: whereas the authors of [1] are keen on reducing the number of additions (good for circuit complexity), we rather optimize the number of machine instructions on recent general purpose CPUs (good for software implementations). Our idea to choose moduli $2^n - \varepsilon$ that can be factored into smaller moduli is new.

Other algorithms for speeding up multiple multi-modular reductions and reconstructions for the same moduli (while allowing for additional precomputations) have recently been proposed in [7]. These algorithms essentially replace all divisions by simple multiplications and can be used in conjunction with our new algorithms for conversions between residues modulo $m_i = m_{i,1} \cdots m_{i,s}$ and residues modulo $m_{i,1}, \ldots, m_{i,s}$.

2 Chinese Remaindering

2.1 The Chinese Remainder Theorem

For any integer $m \geqslant 1$, we recall that $\mathcal{R}_m = \{0, \ldots, m-1\}$. For machine computations, it is convenient to use the following effective version of the Chinese remainder theorem:

Chinese Remainder Theorem. *Let $m_1, \ldots, m_\ell$ be positive integers that are mutually coprime and denote $M = m_1 \cdots m_m$. There exist $c_1, \ldots, c_\ell \in \mathcal{R}_M$ such that for any $a_1 \in \mathcal{R}_{m_1}, \ldots, a_\ell \in \mathcal{R}_{m_\ell}$, the number*

$$x = (c_1 a_1 + \cdots + c_\ell a_\ell) \text{ rem } M$$

satisfies $x \text{ rem } m_i = a_i$ for $i = 1, \ldots, \ell$.

Proof. For each $i = 1, \ldots, \ell$, let $\pi_i = M/m_i$. Since π_i and m_i are coprime, π_i admits an inverse u_i modulo m_i in $\mathcal{R}_{m_i}$. We claim that we may take $c_i = \pi_i u_i$. Indeed, for $x = (c_1 a_1 + \cdots + c_\ell a_\ell) \operatorname{rem} M$ and any $i \in \{1, \ldots, \ell\}$, we have

$$x \equiv a_1 \pi_1 u_1 + \cdots + a_\ell \pi_\ell u_\ell \pmod{m_i}$$

Since π_j is divisible by m_i for all $j \neq i$, this congruence relation simplifies into

$$x \equiv a_i \pi_i u_i \quad \equiv \quad a_i \pmod{m_i}.$$

This proves our claim and the theorem.

Notation. *We call $c_1, \ldots, c_\ell$ the cofactors for $m_1, \ldots, m_\ell$ in M and also denote these numbers by $c_{m_1;M} = c_1, \ldots, c_{m_\ell;M} = c_\ell$.*

2.2 Modular Arithmetic

For practical computations, the moduli m_i are usually chosen such that they fit into single machine words. Let μ denote the bitsize of a machine word, so that we typically have $\mu = 32$ or $\mu = 64$. It depends on specifics of the processor how basic arithmetic operations modulo m_i can be implemented most efficiently.

For instance, some processors have instructions for multiplying two μ-bit integers and return the exact (2μ)-bit product. If not, then we rather have to assume that the moduli m_i fit into $\mu/2$ instead of μ bits, or replace μ by $\mu/2$. Some processors do not provide efficient integer arithmetic at all. In that case, one might rather wish to rely on floating point arithmetic and take $\mu = 52$ (assuming that we have hardware support for double precision). For floating point arithmetic it also matters whether the processor offers a "fused-multiply-add" (FMA) instruction; this essentially provides us with an efficient way to multiply two μ-bit integers exactly using floating point arithmetic.

It is also recommended to choose moduli m_i that fit into slightly less than μ bits whenever possible. Such extra bits can be used to significantly accelerate implementations of modular arithmetic. For a more detailed survey of practically efficient algorithms for modular arithmetic, we refer to [12].

2.3 Naive Multi-modular Reduction and Reconstruction

Let $m_1, \ldots, m_\ell$, $M = m_1 \cdots m_\ell$, $a_1 \in \mathcal{R}_{m_1}, \ldots, a_\ell \in \mathcal{R}_{m_\ell}$ and $x \in \mathcal{R}_M$ be as in the Chinese remainder theorem. We will refer to the computation of $a_1, \ldots, a_\ell$ as a function of x as the problem of *multi-modular reduction*. The inverse problem is called *multi-modular reconstruction*. In what follows, we assume that $m_1, \ldots, m_\ell$ have been fixed once and for all.

The simplest way to perform multi-modular reduction is to simply take

$$a_i := x \operatorname{rem} m_i \qquad (i = 1, \ldots, \ell). \tag{1}$$

Inversely, the Chinese remainder theorem provides us with a formula for multi-modular reconstruction:

$$x := (c_{m_1;M} a_1 + \cdots + c_{m_\ell;M} a_\ell) \text{ rem } M. \tag{2}$$

Since $m_1, \ldots, m_\ell$ are fixed, the computation of the cofactors $c_{m_1;M}$ can be regarded as a precomputation.

Assume that our hardware provides an instruction for the exact multiplication of two integers that fit into a machine word. If m_i fits into a machine word, then so does the remainder $a_i = x \text{ rem } m_i$. Cutting $c_{m_i;M}$ into ℓ machine words, it follows that the product $c_{m_i;M} a_i$ can be computed using ℓ hardware products and ℓ hardware additions. Inversely, the Euclidean division of an ℓ-word integer x by m_i can be done using $2\ell + O(1)$ multiplications and $2\ell + O(1)$ additions/subtractions: we essentially have to multiply the quotient by m_i and subtract the result from x; each next word of the quotient is obtained through a one word multiplication with an approximate inverse of m_i.

The above analysis shows that the naive algorithm for multi-modular reduction of x modulo $m_1, \ldots, m_\ell$ requires $2\ell^2 + O(\ell)$ hardware multiplications and $2\ell^2 + O(\ell)$ additions. The multi-modular reconstruction of x rem M can be done using only $\ell^2 + O(\ell)$ multiplications and $\ell^2 + O(\ell)$ additions. Depending on the hardware, the moduli m_i, and the way we implement things, $O(\ell^2)$ more operations may be required for the carry handling—but it is beyond the scope of this paper to go into this level of detail.

3 Gentle Moduli

3.1 The Naive Algorithms Revisited for Special moduli

Let us now reconsider the naive algorithms from Sect. 2.3, but in the case when the moduli $m_1, \ldots, m_\ell$ are all close to a specific power of two. More precisely, we assume that

$$m_i = 2^{sw} + \delta_i \qquad (i = 1, \ldots, \ell), \tag{3}$$

where $|\delta_i| \leqslant 2^{w-1}$ and $s \geqslant 2$ a small number. As usual, we assume that the m_i are pairwise coprime and we let $M = m_1 \cdots m_\ell$. We also assume that w is slightly smaller than μ and that we have a hardware instruction for the exact multiplication of μ-bit integers.

For moduli m_i as in (3), the naive algorithm for the Euclidean division of a number $x \in \mathcal{R}_{2^{\ell sw}}$ by m_i becomes particularly simple and essentially boils down to the multiplication of δ_i with the quotient of this division. In other words, the remainder can be computed using $\sim \ell s$ hardware multiplications. In comparison, the algorithm from Sect. 2.3 requires $\sim 2\ell s^2$ multiplication when applied to (sw)-bit (instead of w-bit) integers. More generally, the computation of ℓ remainders $a_1 = x \text{ rem } m_1, \ldots, a_\ell = x \text{ rem } m_\ell$ can be done using $\sim \ell^2 s$ instead of $\sim 2\ell^2 s^2$ multiplications. This leads to a potential gain of a factor $2s$,

although the remainders are (sw)-bit integers instead of w-bit integers, for the time being.

Multi-modular reconstruction can also be done faster, as follows, using similar techniques as in [1,5]. Let $x \in \mathcal{R}_M$. Besides the usual binary representation of x and the multi-modular representation $(a_1, \ldots, a_\ell) = (x \text{ rem } m_1, \ldots, x \text{ rem } m_\ell)$, it is also possible to use the *mixed radix representation* (or *Newton representation*)

$$x = b_1 + b_2 m_1 + b_3 m_1 m_2 + \cdots + b_\ell m_1 \cdots m_{\ell-1},$$

where $b_i \in \mathcal{R}_{m_i}$. Let us now show how to obtain $(b_1, \ldots, b_\ell)$ efficiently from $(a_1, \ldots, a_\ell)$. Since $x \text{ rem } m_1 = b_1 = a_1$, we must take $b_1 = a_1$. Assume that $b_1, \ldots, b_{i-1}$ have been computed. For $j = i-1, \ldots, 1$ we next compute

$$u_{j,i} = (b_j + b_{j+1} m_j + \cdots + b_{i-1} m_j \cdots m_{i-2}) \text{ rem } m_i$$

using $u_{i-1,i} = b_{i-1}$ and

$$\begin{aligned} u_{j,i} &= (b_j + u_{j+1,i} m_j) \text{ rem } m_i \\ &= (b_j + u_{j+1,i} \cdot (\delta_j - \delta_i)) \text{ rem } m_i \qquad (j = i-2, \ldots, 1). \end{aligned}$$

Notice that $u_{i-1,i}, \ldots, u_{1,i}$ can be computed using $(i-1)(s+1)$ hardware multiplications. We have

$$x \text{ rem } m_i = (u_{1,i} + b_i m_1 \cdots m_{i-1}) \text{ rem } m_i \;=\; a_i.$$

Now the inverse v_i of $m_1 \cdots m_{i-1}$ modulo m_i can be precomputed. We finally compute

$$b_i = v_i (a_i - u_{1,i}) \text{ rem } m_i,$$

which requires $s^2 + O(s)$ multiplications. For small values of i, we notice that it may be faster to divide successively by $m_1, \ldots, m_{i-1}$ modulo m_i instead of multiplying with v_i. In total, the computation of the mixed radix representation $(b_1, \ldots, b_\ell)$ can be done using $\binom{\ell}{2}(s+1) + \ell s^2 + O(\ell s)$ multiplications. Having computed the mixed radix representation, we next compute

$$x_i = b_i + b_{i+1} m_i + \cdots + b_\ell m_i \cdots m_{\ell-1}$$

for $i = \ell, \ldots, 1$, using the recurrence relation

$$x_i = b_i + x_{i+1} m_i.$$

Since $x_{i+1} \in \mathcal{R}_{2^{(\ell-i)sw}}$, the computation of x_i requires $(\ell - i)s$ multiplications. Altogether, the computation of $x = x_1$ from $(a_1, \ldots, a_\ell)$ can therefore be done using $\binom{\ell}{2}(2s+1) + \ell s^2 \approx \ell s(\ell + s)$ hardware multiplications.

3.2 Combining Special Moduli into Gentle moduli

For practical applications, we usually wish to work with moduli that fit into one word (instead of s words). With the m_i as in the previous subsection, this means that we need to impose the further requirement that each modulus m_i can be factored

$$m_i = m_{i,1} \cdots m_{i,s},$$

with $m_{i,1}, \ldots, m_{i,s} < 2^\mu$. If this is possible, then the m_i are called *s-gentle moduli*. For given bitsizes w and $s \geqslant 2$, the main questions are now: do such moduli indeed exist? If so, then how to find them?

If $s = 2$, then it is easy to construct s-gentle moduli $m_i = 2^{2w} + \delta_i$ by taking $\delta_i = -\varepsilon_i^2$, where $0 \leqslant \varepsilon_i < 2^{(w-1)/2}$ is odd. Indeed,

$$2^{2w} - \varepsilon_i^2 = (2^w + \varepsilon_i)(2^w - \varepsilon_i)$$

and $\gcd(2^w + \varepsilon_i, 2^w - \varepsilon_i) = \gcd(2^w + \varepsilon_i, 2\varepsilon_i) = \gcd(2^w + \varepsilon_i, \varepsilon_i) = \gcd(2^w, \varepsilon_i) = 1$. Unfortunately, this trick does not generalize to higher values $s \geqslant 3$. Indeed, consider a product

$$\begin{aligned}(2^w + \eta_1) \cdots (2^w + \eta_s) = 2^{sw} + (\eta_1 + \cdots + \eta_s)2^{(s-1)w} + \\ ((\eta_1 + \cdots + \eta_s)^2 - (\eta_1^2 + \cdots + \eta_s^2))2^{(s-2)w-1} + \cdots,\end{aligned}$$

where $\eta_1, \ldots, \eta_s$ are small compared to 2^w. If the coefficient $\eta_1 + \cdots + \eta_s$ of $2^{(s-1)w}$ vanishes, then the coefficient of $2^{(s-2)w-1}$ becomes the opposite $-(\eta_1^2 + \cdots + \eta_s^2)$ of a sum of squares. In particular, both coefficients cannot vanish simultaneously, unless $\eta_1 = \cdots = \eta_s = 0$.

If $s > 2$, then we are left with the option to search s-gentle moduli by brute force. As long as s is "reasonably small" (say $s \leqslant 8$), the probability to hit an s-gentle modulus for a randomly chosen δ_i often remains significantly larger than 2^{-w}. We may then use sieving to find such moduli. By what precedes, it is also desirable to systematically take $\delta_i = -\varepsilon_i^2$ for $0 \leqslant \varepsilon_i < 2^{(w-1)/2}$. This has the additional benefit that we "only" have to consider $2^{(w-1)/2}$ possibilities for ε_i.

We will discuss sieving in more detail in the next section. Assuming that we indeed have found s-gentle moduli $m_1, \ldots, m_\ell$, we may use the naive algorithms from Sect. 2.3 to compute $(x \text{ rem } m_{i,1}, \ldots, x \text{ rem } m_{i,s})$ from $x \text{ rem } m_i$ and *vice versa* for $i = 1, \ldots, \ell$. Given $x \text{ rem } m_i$ for all $i = 1, \ldots, \ell$, this allows us to compute all remainders $x \text{ rem } m_{i,j}$ using $2\ell s^2 + O(\ell s)$ hardware multiplications, whereas the opposite conversion only requires $\ell s^2 + O(\ell s)$ multiplications. Altogether, we may thus obtain the remainders $x \text{ rem } m_{i,j}$ from $x \text{ rem } M$ and *vice versa* using $\sim \ell s(\ell + 2s)$ multiplications.

4 The Gentle Modulus Hunt

4.1 The Sieving Procedure

We implemented a sieving procedure in MATHEMAGIX [11] that uses the MPARI package with an interface to PARI-GP [15]. Given parameters s, w, w' and μ, the

goal of our procedure is to find s-gentle moduli of the form

$$M = (2^{sw/2} + \varepsilon)(2^{sw/2} - \varepsilon) \quad = \quad m_1 \cdots m_s$$

with the constraints that

$$m_i < 2^{w'}$$
$$\gcd(m_i, 2^\mu!) = 1,$$

for $i = 1, \ldots, s$, and $m_1 \leqslant \cdots \leqslant m_s$. The parameter s is small and even. One should interpret w and w' as the intended and maximal bitsize of the small moduli m_i. The parameter μ stands for the minimal bitsize of a prime factor of m_i. The parameter ε should be such that $4\varepsilon^2$ fits into a machine word.

In Table 1 below we have shown some experimental results for this sieving procedure in the case when $s = 6$, $w = 22$, $w' = 25$ and $\mu = 4$. For $\varepsilon < 1000000$, the table provides us with ε, the moduli $m_1, \ldots, m_s$, as well as the smallest prime power factors of the product M. Many hits admit small prime factors, which increases the risk that different hits are not coprime. For instance, the number 17 divides both $2^{132} - 311385^2$ and $2^{132} - 376563^2$, whence these 6-gentle moduli cannot be selected simultaneously (except if one is ready to sacrifice a few bits by working modulo $\operatorname{lcm}(2^{132} - 311385^2, 2^{132} - 376563^2)$ instead of $(2^{132} - 311385^2) \cdot (2^{132} - 376563^2)$).

In the case when we use multi-modular arithmetic for computations with rational numbers instead of integers (see [9, Sect. 5 and, more particularly, Sect. 5.10]), then small prime factors should completely be prohibited, since they increase the probability of divisions by zero. For such applications, it is therefore desirable that $m_1, \ldots, m_s$ are all prime. In our table, this occurs for $\varepsilon = 57267$ (we indicated this by highlighting the list of prime factors of M).

In order to make multi-modular reduction and reconstruction as efficient as possible, a desirable property of the moduli m_i is that they either divide $2^{sw/2} - \varepsilon$ or $2^{sw/2} + \varepsilon$. In our table, we highlighted the ε for which this happens. We notice that this is automatically the case if $m_1, \ldots, m_s$ are all prime. If only a small number of m_i (say a single one) do not divide either $2^{sw/2} - \varepsilon$ or $2^{sw/2} + \varepsilon$, then we remark that it should still be possible to design reasonably efficient *ad hoc* algorithms for multi-modular reduction and reconstruction.

Another desirable property of the moduli $m_1 \leqslant \cdots \leqslant m_s$ is that m_s is as small as possible: the spare bits can for instance be used to speed up matrix multiplication modulo m_s. Notice however that one "occasional" large modulus m_s only impacts on one out of s modular matrix products; this alleviates the negative impact of such moduli. We refer to Sect. 4.3 below for more details.

For actual applications, one should select gentle moduli that combine all desirable properties mentioned above. If not enough such moduli can be found, then it depends on the application which criteria are most important and which ones can be released.

4.2 Influence of the Parameters s, w and w'

Ideally speaking, we want s to be as large as possible. Furthermore, in order to waste as few bits as possible, w' should be close to the word size (or half of it) and $w' - w$ should be minimized. When using double precision floating point arithmetic, this means that we wish to take $w' \in \{24, 25, 26, 50, 51, 52\}$. Whenever we have efficient hardware support for integer arithmetic, then we might prefer $w \in \{30, 31, 32, 62, 63, 64\}$.

Let us start by studying the influence of $w' - w$ on the number of hits. In Table 2, we have increased w by one with respect to Table 1. This results in an approximate 5% increase of the "capacity" sw of the modulus M. On the one hand, we observe that the hit rate of the sieve procedure roughly decreases by a factor of thirty. On the other hand, we notice that the rare gentle moduli that we do find are often of high quality (on four occasions the moduli $m_1, \ldots, m_s$ are all prime in Table 2).

Without surprise, the hit rate also sharply decreases if we attempt to increase s. The results for $s = 8$ and $w = 22$ are shown in Table 3. A further infortunate side effect is that the quality of the gentle moduli that we do find also decreases. Indeed, on the one hand, M tends to systematically admit at least one small prime factor. On the other hand, it is rarely the case that each m_i divides either $2^{sw/2} - \varepsilon$ or $2^{sw/2} + \varepsilon$ (this might nevertheless happen for other recombinations of the prime factors of M, but only modulo a further increase of m_s).

An increase of w' while maintaining s and $w' - w$ fixed also results in a decrease of the hit rate. Nevertheless, when going from $w' = 25$ (floating point

Table 1. List of 6-gentle moduli for $w = 22$, $w' = 25$, $\mu = 4$ and $\varepsilon < 1000000$.

ε	m_1	m_2	m_3	m_4	m_5	m_6	$p_1^{\nu_1}, p_2^{\nu_2}, \ldots$
27657	28867	4365919	6343559	13248371	20526577	25042063	$29, 41, 43, 547, \ldots$
57267	416459	1278617	2041469	6879443	25754563	28268089	$416459, \ldots$
77565	7759	8077463	8261833	18751793	19509473	28741799	$59, 641, \ldots$
95253	724567	965411	3993107	4382527	19140643	23236813	$43, 724567, \ldots$
294537	190297	283729	8804561	19522819	19861189	29537129	$23^2, 151, 1879, \ldots$
311385	145991	4440391	4888427	6812881	7796203	32346631	$17, 79, 131, \ldots$
348597	114299	643619	6190673	11389121	32355397	32442427	$31, 277, \ldots$
376563	175897	1785527	2715133	7047419	30030061	30168739	$17, 127, 1471, \ldots$
462165	39841	3746641	7550339	13195943	18119681	20203643	$67, 641, 907, \ldots$
559713	353201	873023	2595031	11217163	18624077	32569529	$19, 59, 14797, \ldots$
649485	21727	1186571	14199517	15248119	31033397	31430173	$19, 109, 227, \ldots$
656997	233341	1523807	5654437	8563679	17566069	18001723	$79, 89, 63533, \ldots$
735753	115151	923207	3040187	23655187	26289379	27088541	$53, 17419, \ldots$
801687	873767	1136111	3245041	7357871	8826871	26023391	$23, 383777, \ldots$
826863	187177	943099	6839467	11439319	12923753	30502721	$73, 157, 6007, \ldots$
862143	15373	3115219	11890829	18563267	19622017	26248351	$31, 83, 157, \ldots$
877623	514649	654749	4034687	4276583	27931549	33525223	$41, 98407, \ldots$
892455	91453	2660297	3448999	12237457	21065299	25169783	$29, 397, 2141, \ldots$

Table 2. List of 6-gentle moduli for $w = 23$, $w' = 25$, $\mu = 4$ and $\varepsilon < 16000000$.

ε	m_1	m_2	m_3	m_4	m_5	m_6	$p_1^{\nu_1}, p_2^{\nu_2}, \ldots$
936465	543889	4920329	12408421	15115957	24645539	28167253	$19, 59, 417721, \ldots$
2475879	867689	4051001	11023091	13219163	24046943	28290833	$867689, \ldots$
3205689	110161	12290741	16762897	22976783	25740731	25958183	$59, 79, 509, \ldots$
3932205	4244431	5180213	5474789	8058377	14140817	25402873	$4244431, \ldots$
5665359	241739	5084221	18693097	21474613	23893447	29558531	$31, 41, 137, \ldots$
5998191	30971	21307063	21919111	22953967	31415123	33407281	$101, 911, 941, \ldots$
6762459	3905819	5996041	7513223	7911173	8584189	29160587	$43, 137, 90833, \ldots$
9245919	2749717	4002833	8274689	9800633	15046937	25943587	$2749717, \ldots$
9655335	119809	9512309	20179259	21664469	22954369	30468101	$17, 89, 149, \ldots$
12356475	1842887	2720359	7216357	13607779	23538769	30069449	$1842887, \ldots$
15257781	1012619	5408467	9547273	11431841	20472121	28474807	$31, 660391, \ldots$

Table 3. List of 8-gentle moduli for $w = 22$, $w' = 25$, $\mu = 4$ and $\varepsilon < 10000000$.

ε	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	$p_1^{\nu_1}, p_2^{\nu_2}, \ldots$
889305	50551	1146547	4312709	5888899	14533283	16044143	16257529	17164793	$17, 31, 31, 59, \ldots$
2447427	53407	689303	3666613	4837253	7944481	21607589	25976179	32897273	$31, 61, 103, \ldots$
2674557	109841	1843447	2624971	5653049	7030883	8334373	18557837	29313433	$103, 223, 659, \ldots$
3964365	10501	2464403	6335801	9625841	10329269	13186219	17436197	25553771	$23, 163, 607, \ldots$
4237383	10859	3248809	5940709	6557599	9566959	11249039	22707323	28518509	$23, 163, 1709, \ldots$
5312763	517877	616529	879169	4689089	9034687	11849077	24539909	27699229	$43, 616529, \ldots$
6785367	22013	1408219	4466089	7867589	9176941	12150997	26724877	29507689	$23, 41, 197, \ldots$
7929033	30781	730859	4756351	9404807	13807231	15433939	19766077	22596193	$31, 307, 503, \ldots$
8168565	10667	3133103	3245621	6663029	15270019	18957559	20791819	22018021	$43, 409, 467, \ldots$
8186205	41047	2122039	2410867	6611533	9515951	14582849	16507739	30115277	$23, 167, 251, \ldots$

arithmetic) to $w' = 31$ (integer arithmetic), this is counterbalanced by the fact that ε can also be taken larger (namely $\varepsilon < 2^{w'}$); see Table 4 for a concrete example. When doubling w and w' while keeping the same upper bound for ε, the hit rate remains more or less unchanged, but the rate of high quality hits tends to decrease somewhat: see Table 5.

It should be possible to analyze the hit rate as a function of the parameters s, w, w' and μ from a probabilistic point of view, using the idea that a random number n is prime with probability $(\log n)^{-1}$. However, from a practical perspective, the priority is to focus on the case when $w' \leqslant 64$. For the most significant choices of parameters $\mu < w < w' \leqslant 64$ and s, it should be possible to compile full tables of s-gentle moduli. Unfortunately, our current implementation is still somewhat inefficient for $w' > 32$. A helpful feature for upcoming versions of Pari would be a function to find all prime factors of an integer below a specified maximum $2^{w'}$ (the current version only does this for prime factors that can be tabulated).

Table 4. List of 6-gentle moduli for $w = 28$, $w' = 31$, $\mu = 4$ and $\varepsilon < 1600000$. Followed by some of the next gentle moduli for which each m_i divides either $2^{sw/2} - \alpha$ or $2^{sw/2} + \alpha$.

ε	m_1	m_2	m_3	m_4	m_5	m_6	$p_1^{\nu_1}, p_2^{\nu_2}, \ldots$
303513	42947057	53568313	331496959	382981453	1089261409	1176003149	$29^2, 1480933, \ldots$
851463	10195123	213437143	470595299	522887483	692654273	1008798563	$17, 41, 67, \ldots$
1001373	307261	611187931	936166801	1137875633	1196117147	1563634747	$47, 151, \ldots$
1422507	3950603	349507391	490215667	684876553	693342113	1164052193	$29, 211, 349, \ldots$
1446963	7068563	94667021	313871791	877885639	1009764377	2009551553	$23, 71, 241, \ldots$
1551267	303551	383417351	610444753	1178193077	2101890797	2126487631	$29, 43, 2293, \ldots$
1555365	16360997	65165071	369550981	507979403	1067200639	1751653069	$17, 23, 67, \ldots$
4003545	20601941	144707873	203956547	624375041	655374931	1503716491	$47, 67, \ldots$
4325475	11677753	139113383	210843443	659463289	936654347	1768402001	$19, 41, \ldots$
4702665	8221903	131321017	296701997	496437899	1485084431	1584149417	$8221903, \ldots$
5231445	25265791	49122743	433700843	474825677	907918279	1612324823	$17, 1486223, \ldots$
5425527	37197571	145692101	250849363	291039937	456174539	2072965393	$37197571, \ldots$
6883797	97798097	124868683	180349291	234776683	842430863	858917923	$97798097, \ldots$
7989543	4833137	50181011	604045619	638131951	1986024421	2015143349	$23, 367, \ldots$

Table 5. List of 6-gentle moduli for $w = 44$, $w' = 50$, $\mu = 4$ and $\varepsilon < 200000$. Followed by some of the next gentle moduli for which each m_i divides either $2^{sw/2} - \alpha$ or $2^{sw/2} + \alpha$.

ε	m_1	m_2	$\cdots$	m_5	m_6	$p_1^{\nu_1}, p_2^{\nu_2}, \ldots$
15123	380344780931	774267432193	$\cdots$	463904018985637	591951338196847	$37, 47, 239, \ldots$
34023	9053503517	13181369695139	$\cdots$	680835893479031	723236090375863	$29, 35617, \ldots$
40617	3500059133	510738813367	$\cdots$	824394263006533	1039946916817703	$23, 61, 347, \ldots$
87363	745270007	55797244348441	$\cdots$	224580313861483	886387548974947	$71, 9209, \ldots$
95007	40134716987	2565724842229	$\cdots$	130760921456911	393701833767607	$19, 67, \ldots$
101307	72633113401	12070694419543	$\cdots$	95036720090209	183377870340761	$41, 401, \ldots$
140313	13370367761	202513228811	$\cdots$	397041457462499	897476961701171	$379, 1187, \ldots$
193533	35210831	15416115621749	$\cdots$	727365428298107	770048329509499	$59, 79, \ldots$
519747	34123521053	685883716741	$\cdots$	705516472454581	836861326275781	$127, 587, \ldots$
637863	554285276371	1345202287357	$\cdots$	344203886091451	463103013579761	$79, 1979, \ldots$
775173	322131291353	379775454593	$\cdots$	194236314135719	1026557288284007	$322131291353, \ldots$
913113	704777248393	1413212491811	$\cdots$	217740328855369	261977228819083	$37, 163, 677, \ldots$
1400583	21426322331	42328735049	$\cdots$	411780268096919	626448556280293	$21426322331, \ldots$

4.3 Application to Matrix Multiplication

Let us finally return to our favourite application of multi-modular arithmetic to the multiplication of integer matrices $A, B \in \mathbb{Z}^{r \times r}$. From a practical point of view, the second step of the algorithm from the introduction can be implemented very efficiently if rm_i^2 fits into the size of a word.

When using floating point arithmetic, this means that we should have $rm_i^2 < 2^{52}$ for all i. For large values of r, this is unrealistic; in that case, we subdivide the $r \times r$ matrices into smaller $r_i \times r_i$ matrices with $r_i m_i^2 < 2^{52}$. The fact that r_i may depend on i is very significant. First of all, the larger we can take r_i, the faster we can multiply matrices modulo m_i. Secondly, the m_i in the tables from the previous sections often vary in bitsize. It frequently happens that we may take

all r_i large except for the last modulus m_ℓ. The fact that matrix multiplications modulo the worst modulus m_ℓ are somewhat slower is compensated by the fact that they only account for one out of every ℓ modular matrix products.

Several of the tables in the previous subsections were made with the application to integer matrix multiplication in mind. Consider for instance the modulus $M = m_1 \cdots m_6 = 2^{132} - 656997^2$ from Table 1. When using floating point arithmetic, we obtain $r_1 \leqslant 82713$, $r_2 \leqslant 1939$, $r_3 \leqslant 140$, $r_4 \leqslant 61$, $r_5 \leqslant 14$ and $r_6 \leqslant 13$. Clearly, there is a trade-off between the efficiency of the modular matrix multiplications (high values of r_i are better) and the bitsize $\approx \ell w$ of M (larger capacities are better).

References

1. Bajard, J.C., Kaihara, M.E., Plantard, T.: Selected RNS bases for modular multiplication. In: Proceedings of the 19th IEEE Symposium on Computer Arithmetic, pp. 25–32 (2009)
2. Bernstein, D.: Scaled remainder trees (2004). https://cr.yp.to/arith/scaledmod-20040820.pdf
3. Borodin, A., Moenck, R.T.: Fast modular transforms. J. Comput. Syst. Sci. **8**, 366–386 (1974)
4. Bostan, A., Lecerf, G., Schost, É.: Tellegen's principle into practice. In: Proceedings of ISSAC 2003, pp. 37–44. ACM Press (2003)
5. Bostan, A., Schost, É.: Polynomial evaluation and interpolation on special sets of points. J. Complex. **21**(4), 420–446 (2005). Festschrift for the 70th Birthday of Arnold Schönhage
6. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Math. Comput. **19**, 297–301 (1965)
7. Doliskani, J., Giorgi, P., Lebreton, R., Schost, É.: Simultaneous conversions with the Residue Number System using linear algebra. Technical report, HAL (2016). https://hal-lirmm.ccsd.cnrs.fr/lirmm-01415472
8. Fiduccia, C.M.: Polynomial evaluation via the division algorithm: the fast Fourier transform revisited. In: Rosenberg, A.L. (ed.) Fourth Annual ACM Symposium on Theory of Computing, pp. 88–93 (1972)
9. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra, 3rd edn. Cambridge University Press, New York (2013)
10. van der Hoeven, J.: Faster Chinese remaindering. Technical report, HAL (2016). http://hal.archives-ouvertes.fr/hal-01403810
11. van der Hoeven, J., Lecerf, G., Mourrain, B., et al.: Mathemagix (2002). http://www.mathemagix.org
12. van der Hoeven, J., Lecerf, G., Quintin, G.: Modular SIMD arithmetic in Mathemagix. ACM Trans. Math. Softw. **43**(1), 5:1–5:37 (2016)
13. Karatsuba, A., Ofman, J.: Multiplication of multidigit numbers on automata. Soviet Phys. Doklady **7**, 595–596 (1963)
14. Moenck, R.T., Borodin, A.: Fast modular transforms via division. In: Thirteenth annual IEEE Symposium on Switching and Automata Theory, pp. 90–96, University of Maryland, College Park (1972)
15. The PARI Group, Bordeaux. PARI/GP (2012). Software. http://pari.math.u-bordeaux.fr

Homotopies for Connected Components of Algebraic Sets with Application to Computing Critical Sets

Daniel J. Bates[1], Dani A. Brake[2(✉)], Jonathan D. Hauenstein[3], Andrew J. Sommese[3], and Charles W. Wampler[3]

[1] Department of Mathematics, Colorado State University, Fort Collins, CO 80523, USA
bates@math.colostate.edu
[2] Department of Mathematics, University of Wisconsin - Eau Claire, Eau Claire, WI 54702, USA
brakeda@uwec.edu
[3] Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556, USA
{hauenstein,sommese}@nd.edu, charles.w.wampler@gm.com
http://www.math.colostate.edu/~bates
http://www.danibrake.org
http://www.nd.edu/~jhauenst
http://www.nd.edu/~sommese
http://www.nd.edu/~cwample1

Abstract. Given a polynomial system f, this article provides a general construction for homotopies that yield at least one point of each connected component on the set of solutions of $f = 0$. This algorithmic approach is then used to compute a superset of the isolated points in the image of an algebraic set which arises in many applications, such as computing critical sets used in the decomposition of real algebraic sets. An example is presented which demonstrates the efficiency of this approach.

Keywords: Numerical algebraic geometry · Homotopy continuation Projections

AMS Subject Classification: 65H10 · 68W30 · 14P05

D. J. Bates—Supported in part by AFOSR grant FA8650-13-1-7317, NSF ACI-1440467, and NSF DMS-1719658.

D. A. Brake and J. D. Hauenstein—Supported in part by AFOSR grant FA8650-13-1-7317 and NSF ACI-1460032.

A. J. Sommese—Supported in part by the Duncan Chair of the University of Notre Dame, AFOSR grant FA8650-13-1-7317, and NSF ACI-1440607.

C. W. Wampler—Supported in part by AFOSR grant FA8650-13-1-7317.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 107–120, 2017.
https://doi.org/10.1007/978-3-319-72453-9_8

1 Introduction

For a polynomial system f with complex coefficients, the fundamental problem of algebraic geometry is to understand the set of solutions of the system $f = 0$, denoted $\mathcal{V}(f)$. *Numerical algebraic geometry* (see, e.g., [5,25] for a general overview) is based on using homotopy continuation methods for computing $\mathcal{V}(f)$. Geometrically, one can decompose $\mathcal{V}(f)$ into its irreducible components, which corresponds numerically to computing a numerical irreducible decomposition with each irreducible component represented by a witness set. The first step of computing a numerical irreducible decomposition is to compute witness point supersets with the algorithms [13,22,24] relying upon a sequence of homotopies. At each dimension where a solution component could exist, a generic linear space of complementary dimension is used to slice the solution set; the witness points are then the isolated points in the intersection of the solution component and the linear slice. Accordingly, a crucial property of the algorithms employed is that they must generate a finite set of points, say S, in the slice that includes all isolated points of the slice.

In this article, we change the focus from irreducible components to connected components. We present an approach that computes a finite set of points in $\mathcal{V}(f)$ containing at least one point on each connected component of $\mathcal{V}(f)$ using a single homotopy, built on a similar theoretical viewpoint as the nonconstructive approach presented in [19, Theorem 7]. This work is complementary to methods for computing a finite set of points in the set of real points in $\mathcal{V}(f)$, denoted $\mathcal{V}_{\mathbb{R}}(f)$, containing at least one point on each connected component of $\mathcal{V}_{\mathbb{R}}(f)$ [1,11,21,30].

Our approach is particularly relevant to *numerical elimination theory* [5, Chap. 16], which seeks to treat projections of algebraic sets in a similar fashion as general algebraic sets but without having on hand polynomials that vanish on the projection (and without computing such polynomials). This is a numerical alternative to symbolic elimination methods [29]. In particular, suppose that $f(x, y)$ is a polynomial system that is defined on a product of two projective spaces, and let $X = \pi(\mathcal{V}(f))$ where $\pi(x, y) = x$. We do not have a polynomial system that defines X, so we do all computations via points in its pre-image, $\pi^{-1}(X) \cap \mathcal{V}(f)$. In particular, if we wish to compute a finite set of points $S \subset \mathcal{V}(f)$ such that $\pi(S)$ includes all isolated points of X, it suffices if S contains a point on each connected component of $\mathcal{V}(f)$. Our new algorithm enables one to compute such a set S using a single homotopy; one does not need to separately consider each possible dimension of the fibers over the isolated points of X.

The viewpoint of computing based on connected components also has many other applications, particularly related to so-called critical point conditions. For example, the methods mentioned above in relation to real solutions, namely [1,11,21,30], compute critical points of $\mathcal{V}(f)$ with respect to the distance function (see also [9]). In [6,7], critical points of $\mathcal{V}(f)$ with respect to a linear projection are used to numerically decompose real algebraic sets. (We discuss this in more detail in Sect. 4.) Other applications include computing witness point sets for irreducible components of rank-deficiency sets [2], isosingular sets [14], and deflation ideals [17].

To highlight the key point of this paper, consider computing rank-deficiency sets as in [2]. With this setup, one adds new variables related to the null space of the matrix. To make sure that all components of the rank-deficiency sets are computed, traditional approaches need to consider all possible dimensions of the null space. The point of this paper is to provide an algorithmic approach by which one only needs to consider the smallest possible null space dimension, thereby simplifying the computation.

The rest of the article is organized as follows. Section 2 derives an algorithmic approach that computes at least one point on every connected component of $\mathcal{V}(f)$ using one homotopy. This is discussed in relation to elimination theory in Sect. 3, while Sect. 4 focuses on computing critical sets of projections of real algebraic sets. An example illustrating this approach and its efficiency is presented in Sect. 5.

2 Construction of Homotopies

The starting point for constructing one homotopy that computes at least one point on each connected component of a solution set of polynomial equations is [19, Theorem 7]. Since this theorem is nonconstructive, we derive an algorithmic approach for performing this computation in Proposition 1 and sketch a proof. We refer to [25] for details regarding algebraic and analytic sets with [19, Appendix] providing a quick introduction to basic results regarding such sets.

Suppose that $\mathcal{E}$ is a complex algebraic vector bundle on an n-dimensional irreducible and reduced complex projective set X. Denote the bundle projection from $\mathcal{E}$ to X by $\pi_{\mathcal{E}}$. A section s of $\mathcal{E}$ is a complex algebraic map $s : X \to \mathcal{E}$ such that $\pi_{\mathcal{E}} \circ s$ is the identity; i.e., for all $x \in X$, $(\pi_{\mathcal{E}} \circ s)(x) = \pi_{\mathcal{E}}(s(x)) = x$.

There is a nonempty Zariski open set $\mathcal{U} \subset X$ over which $\mathcal{E}$ has a trivialization. Using such a trivialization, an algebraic section of $\mathcal{E}$ becomes a system of $\operatorname{rank}(\mathcal{E})$ algebraic functions. In fact, *all* polynomial systems arise in this way and results about special homotopies which track different numbers of paths, e.g., [16,20,26], are based on this interpretation (see also [25, Appendix A]).

Let us specialize this to a concrete situation.

Example 1. Suppose that $X \subset \prod_{j=1}^{r} \mathbb{P}^{n_j}$ is an irreducible and reduced n-dimensional algebraic subset of a product of projective spaces. For example, X could be an irreducible component of a system of multihomogeneous polynomials in the variables

$$z_{1,0}, \ldots, z_{1,n_1}, \ldots, z_{r,0}, \ldots, z_{r,n_r},$$

where $[z_{j,0}, \ldots, z_{j,n_j}]$ are the homogeneous coordinates on the j^{th} projective space, $\mathbb{P}^{n_j}$. Each homogeneous coordinate $z_{j,k}$ has a natural interpretation as a section of the hyperplane section bundle, denoted $\mathcal{L}_{\mathbb{P}^{n_j}}(1)$. The d^{th} power of the hyperplane section bundle is denoted by $\mathcal{L}_{\mathbb{P}^{n_j}}(d)$. A multihomogeneous polynomial defined on $\prod_{j=1}^{r} \mathbb{P}^{n_j}$ with multidegree $(d_1, \ldots, d_r)$ is naturally interpreted as a section of the line bundle

$$\mathcal{L}_{\prod_{j=1}^{r} \mathbb{P}^{n_j}}(d_1, \ldots, d_r) := \otimes_{j=1}^{r} \pi_j^* \mathcal{L}_{\mathbb{P}^{n_j}}(d_j),$$

where $\pi_k : \prod_{j=1}^{r} \mathbb{P}^{n_j} \to \mathbb{P}^{n_k}$ is the product projection onto the k^{th} factor. A system of n multihomogeneous polynomials

$$f := \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \tag{1}$$

where f_i has multidegree $(d_{i,1}, \ldots, d_{i,n_i})$ is interpreted as a section of

$$\mathcal{E} := \bigoplus_{i=1}^{n} \mathcal{L}_{\prod_{j=1}^{r} \mathbb{P}^{n_j}}(d_{i,1}, \ldots, d_{i,r}).$$

The solution set of $f = 0$ is simply the set of zeroes of the section f.

The n^{th} Chern class of $\mathcal{E}$ [8,10], which lies in the $2n^{\text{th}}$ integer cohomology group $H^{2n}(X, \mathbb{Z})$, is denoted by $c_n(\mathcal{E})$. Let $d := c_n(\mathcal{E})[X] \in \mathbb{Z}$, i.e., d denotes the evaluation of $c_n(\mathcal{E})$ on X.

Example 2. Continuing from Example 1, let $c := \sum_{j=1}^{r} n_j - n$ be the codimension of X. Using multi-index notation for $\alpha = (\alpha_1, \ldots, \alpha_r)$ where each $\alpha_i \geq 0$ and $|\alpha| = \sum_{i=1}^{r} \alpha_i$, we can represent X in homology by

$$\sum_{|\alpha|=c} e_\alpha \mathcal{H}^\alpha$$

where $\mathcal{H}_i := \pi_i^{-1}(H_i)$ with hyperplane $H_i \subset \mathbb{P}^{n_i}$ and $\mathcal{H}^\alpha = \mathcal{H}_1^{\alpha_1} \cdots \mathcal{H}_r^{\alpha_r}$. Moreover, $d := c_n(\mathcal{E})[X]$ is simply the multihomogeneous Bézout number of the system of multihomogeneous polynomials restricted to X, i.e., the coefficient of $\prod_{j=1}^{r} z_j^{n_j}$ in the expression

$$\left(\sum_{|\alpha|=c} e_\alpha z^\alpha \right) \cdot \prod_{i=1}^{n} \left(\sum_{j=1}^{r} d_{i,j} z_j \right).$$

In particular, d is simply the number of zeroes of a general section of $\mathcal{E}$ restricted to X.

A vector space V of global sections of $\mathcal{E}$ is said to *span* $\mathcal{E}$ if, given any point $e \in \mathcal{E}$, there is a section $\sigma \in V$ of $\mathcal{E}$ with $\sigma(\pi_{\mathcal{E}}(e)) = e$. We assume that the rank of $\mathcal{E}$ is $n = \dim X$. If V spans $\mathcal{E}$, then Bertini's Theorem asserts that there is a Zariski dense open set $U \subset V$ with the property that, for all $\sigma \in U$, σ has d nonsingular isolated zeroes contained in the smooth points of X, i.e., the graph of σ meets the graph of the identically zero section of $\mathcal{E}$ transversely in d points in the set of smooth points of X.

Let $|V| := (V \setminus \{0\})/\mathbb{C}^*$ be the space of lines through the origin of V. Given a complex analytic vector bundle $\mathcal{E}$ spanned by a vector space of complex analytic sections V, the total space $\mathcal{Z} \subset X \times |V|$ of solution sets of $s \in V$ is

$$\mathcal{Z} := \{(x, s) \in X \times |V| \ : \ s(x) = 0\}. \tag{2}$$

For simplicity, let $p : \mathcal{Z} \to X$ and $q : \mathcal{Z} \to |V|$ denote the maps induced by the product projections $X \times |V| \to X$ and $X \times |V| \to |V|$, respectively.

Since V spans $\mathcal{E}$, the evaluation map

$$X \times V \to \mathcal{E}$$

is surjective so that the kernel is a vector bundle of rank $\dim V - \operatorname{rank}(\mathcal{E})$. Let $\mathcal{K}$ denote the dual of this kernel and $\mathbb{P}(\mathcal{K})$ denote $(\mathcal{K}^* \setminus X)/\mathbb{C}^*$, the space of lines through the vector space fibers of the bundle projection of $\mathcal{K}^* \to X$. The standard convention of denoting $(\mathcal{K}^* \setminus X)/\mathbb{C}^*$ by $\mathbb{P}(\mathcal{K})$ and not $\mathbb{P}(\mathcal{K}^*)$ is convenient in many calculations.

The space $\mathbb{P}(\mathcal{K})$ is easily identified with $\mathcal{Z}$ and the map p is identified with the map $\mathbb{P}(\mathcal{K}) \to X$ induced by the bundle projection. From this identification, we know that $\mathcal{Z}$ is irreducible.

Let $\mathcal{E}$ denote a rank n algebraic vector bundle on a reduced and irreducible projective algebraic set spanned by a vector space V of algebraic sections of $\mathcal{E}$. Suppose that $\sigma \in V$ and $\tau \in V$ have distinct images in $|V|$ and let $\ell := \langle \sigma, \tau \rangle \subset |V|$ denote the unique projective line, i.e., linear $\mathbb{P}^1$, through the images of σ and τ in $|V|$. Letting λ and μ be homogeneous coordinates on ℓ, i.e., spanning sections of $\mathcal{L}_\ell(1)$, we have the section

$$H(x, \lambda, \mu) := \lambda\sigma + \mu\tau \tag{3}$$

of $q^*_{q^{-1}(\ell)}\mathcal{L}_\ell(1) \otimes p^*\mathcal{E}$. Choosing a trivialization of $\mathcal{E}$ over a Zariski open dense set U and a trivialization of $\mathcal{L}_\ell(1)$ over a Zariski open dense set of ℓ, e.g., the set where $\mu \neq 0$, H is naturally interpreted as a homotopy. See Fig. 1 for an illustration.

With this general setup, we are now ready to state a specialization of the nonconstructive result [19, Theorem 7]. The key difference is that this specialization immediately yields a constructive algorithm for computing a finite set of points containing at least one point on each connected component of $\sigma^{-1}(0)$.

Proposition 1. *Let $\mathcal{E}$ denote a rank n algebraic vector bundle over an irreducible and reduced n-dimensional projective algebraic set X. Let V be a vector space of sections of $\mathcal{E}$ that spans $\mathcal{E}$. Assume that $d := c_n(\mathcal{E})[X] > 0$ and $\tau \in V$ which has d nonsingular zeroes all contained in the smooth points of X. Let $\sigma \in V$ be a nonzero section of $\mathcal{E}$, which is not a multiple of τ. Let $\ell = \langle \sigma, \tau \rangle$ and H as in (3). Then, there is a nonempty Zariski open set $\mathcal{Q} \subset \ell$ such that*

1. *the map $q_{\mathcal{Z}_\mathcal{Q}}$ of $\mathcal{Z}_\mathcal{Q} := \left\{H^{-1}(0) \cap (X \times \mathcal{Q})\right\}$ to ℓ is d-to-one; and*
2. *the finite set $\overline{\mathcal{Z}_\mathcal{Q}} \cap \sigma^{-1}(0)$ contains at least one point of every connected subset of $\sigma^{-1}(0)$.*

Proof. Let $\mathcal{Z}$ as in (2). The projection map $q : \mathcal{Z} \to |V|$ may be Stein factorized [25, Theorem A.4.8] as $q = s \circ r$ where $r : \mathcal{Z} \to Y$ is an algebraic map with connected fibers onto an algebraic set Y and $s : Y \to |V|$ is an algebraic map with finite fibers. The surjectivity of q implies that s is surjective and $\dim Y = \dim |V|$. Since $\mathcal{Z}$ is irreducible, Y is irreducible.

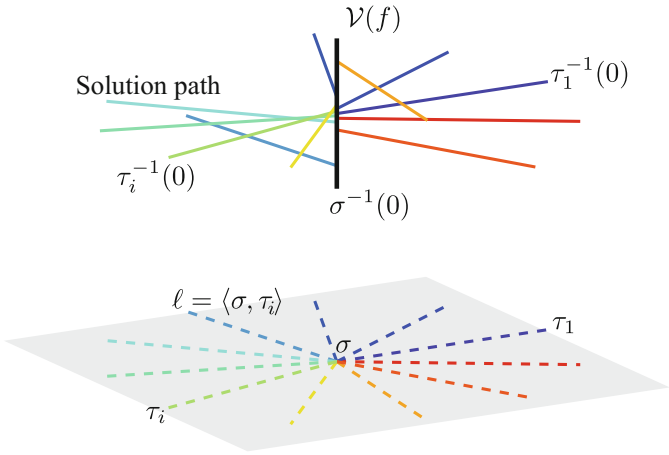


Fig. 1. Illustration of the terminology of the paper. The upper space is in terms of the variables of the problem, with solid lines representing solutions paths, starting at the finite nonsingular zeros of some τ_i, and ending at some zero of σ. We show here many τ_i systems, which all are deformed into σ. At the bottom, the patch represents the vector space V and the lines ℓ interpolate from some τ_i to σ.

It suffices to show that given any $y \in Y$, there is a complex open neighborhood U of y with $s(U)$ an open neighborhood of $s(y)$. A line $\ell \subset |V|$ is defined by $\dim |V| - 1$ linear equations. Thus, $s^{-1}(\ell)$ has all components of dimension at least 1. The result follows from [25, Theorem A.4.17].

Remark 1. If X is a codimension c irreducible component of multiplicity one of the solution set of a polynomial system $f_1, \ldots, f_c$ in the total space, we can choose our homotopy so that the paths over $(0, 1]$ are in the set where $df_1 \wedge \cdots \wedge df_c$ is non-zero.

3 Isolated Points of Images

With the theoretical foundation presented in Sect. 2, this section focuses on computing a finite set of points containing at least one point on each connected component in the image of an algebraic set which, in particular, provides a finite superset of the isolated points in the image. Without loss of generality, it suffices to consider projections of algebraic sets which corresponds algebraically with computing solutions of an elimination ideal.

Lemma 1. *Let V be a closed algebraic subset of a complex quasiprojective algebraic set X. Let $\pi : X \to Y$ denote a proper algebraic map from X to a complex quasiprojective algebraic set Y. If S is a finite set of points in V that contains a point on each connected component of V, then $\pi(S)$ is a finite set of points in $\pi(V)$ which contains a point on each connected component of $\pi(V)$. In particular, $\pi(S)$ is a finite superset of the zero-dimensional components of $\pi(V)$.*

Proof. The image of a connected set under a proper algebraic map is connected.

Consider the concrete case where f is a polynomial system defined on $\mathbb{C}^N \times \mathbb{P}^M$. Let $\mathcal{V}(f) \subset \mathbb{C}^N \times \mathbb{P}^M$ and $Z(f) \subset \mathbb{P}^N \times \mathbb{P}^M$ be the closure of $\mathcal{V}(f)$ under the natural embedding of $\mathbb{C}^N$ into $\mathbb{P}^N$. The approach of Proposition 1 provides one homotopy which can be used to compute a point on each connected component of $Z(f)$. However, it may happen that a point computed on each connected component of $\mathcal{V}(f)$ is at "infinity." One special case is the following for isolated points in the projection of $\mathcal{V}(f)$ onto $\mathbb{C}^N$.

Corollary 1. *Let f be a polynomial system defined on $\mathbb{C}^N \times \mathbb{P}^M$ and π denote the projection $\mathbb{C}^N \times \mathbb{P}^M \to \mathbb{C}^N$. By considering the natural inclusion of $\mathbb{C}^N$ into $\mathbb{P}^N$, let $Z(f)$ be the closure of $\mathcal{V}(f)$ in $\mathbb{P}^N \times \mathbb{P}^M$. Let S be a finite set of points in $Z(f)$ which contains a point on each connected component of $Z(f)$ and $S_{\mathbb{C}} = S \cap (\mathbb{C}^N \times \mathbb{P}^M)$. Then, $\pi(S_{\mathbb{C}})$ is a finite set of points in $\pi(\mathcal{V}(f))$ which contains the isolated points in $\pi(\mathcal{V}(f))$.*

Proof. Suppose that $x \in \pi(\mathcal{V}(f)) \subset \mathbb{C}^N$ is isolated. Let $y \in \mathbb{P}^M$ such that $(x, y) \in \mathcal{V}(f)$. By abuse of notation, we have $(x, y) \in Z(f)$ so that there is a connected component, say C, of $Z(f)$ which contains (x, y). Since x is isolated in $\pi(\mathcal{V}(f))$, we must have $C \subset \{x\} \times \mathbb{P}^M$. The statement follows from the fact that C is thus naturally contained in $\mathbb{C}^N \times \mathbb{P}^M$.

Example 3. To illustrate, consider the polynomial system

$$F(x) = \begin{bmatrix} F_1(x) \\ F_2(x) \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2^2 + x_3^2 + x_4^2 \\ x_1^3 + x_2^3 + x_3^3 + x_4^2 \end{bmatrix}$$

defined on $\mathbb{C}^4$. The set $\mathcal{V}(F) \subset \mathbb{C}^4$ is an irreducible surface of degree six containing one real point, namely the origin, which is an isolated singularity. Since the total derivatives dF_1 and dF_2 are linearly dependent at a singular point, we can consider the following system defined on $\mathbb{C}^4 \times \mathbb{P}^1$:

$$G(x, v) = \begin{bmatrix} F(x) \\ v_0 \cdot dF_1(x) + v_1 \cdot dF_2(x) \end{bmatrix}.$$

Since G consists of 6 polynomials defined on a 5 dimensional space, we reduce to a square system via randomization[1] which, for example, yields:

$$f(x,v) := \begin{bmatrix} x_1^2+x_2^2+x_3^2+x_4^2 \\ x_1^3+x_2^3+x_3^3+x_4^2 \\ v_0(x_1+x_4)+v_1(3x_1^2+x_4) \\ v_0(x_2+x_4)+v_1(3x_2^2+x_4) \\ v_0(x_3+x_4)+v_1(3x_3^2+x_4) \end{bmatrix}.$$

Consider the linear product [26] system:

$$g(x,v) := \begin{bmatrix} x_1^2+x_2^2+x_3^2+x_4^4 \\ x_1^3+x_2^3+x_3^3+x_4^2 \\ (v_0+v_1)(x_1-4x_4-1)(x_1-2) \\ (v_0-v_1)(x_2+2x_4-1)(x_2-3) \\ (v_0+2v_1)(x_3-3x_4-1)(x_3-4) \end{bmatrix}$$

together with the homotopy

$$H((x,v),[\lambda,\mu]) = \lambda f(x,v) + \mu g(x,v).$$

The symbols λ and μ are spanning sections from (3); in this context, they are scalar values interpolating between f and g, and the homotopy "path" variables. With this setup, $g^{-1}(0)$ has exactly $d = 72$ nonsingular isolated solutions which can be computed easily. Further, 72 is the coefficient of a^4b in the polynomial $(2a)(3a)(2a+b)^3$, one way to compute the 2-homogeneous root count [20].

We used `Bertini` [4] to track the 72 paths along a real arc contained in the line $\langle \sigma, \tau \rangle$ in which 30 paths diverge to infinity and 42 paths end at finite points. Of the latter, 20 endpoints are nonsingular isolated solutions which are extraneous in that they arose from the randomization and not actually in $\mathcal{V}(G)$. The other 22 paths converged to points in $\{0\} \times \mathbb{P}^1$: 18 of which ended with $v = [0,1] \in \mathbb{P}^1$ while the other 4 break into 2 groups of 2 with v of the form $[1,\alpha]$ and $[1,\mathrm{conj}(\alpha)]$ where $\alpha \approx -0.351 + 0.504 \cdot \sqrt{-1}$. In particular, even though $\{0\} \times \mathbb{P}^1$ is a positive-dimensional solution component of $\mathcal{V}(f)$ and also of $\mathcal{V}(G)$, we always obtain at least one point on this component showing that the origin is the only point in $\mathcal{V}(F)$ which is singular with respect to F.

[1] In usual practice, "randomization" means replacing a set of polynomials with some number of random linear combinations of the polynomials. When the appropriate number of combinations is used, then in a Zariski-open subset of the Cartesian space of coefficients of the linear combinations, the solution set of interest is preserved. See, for example, [25, Sect. 13.5]. Here, for simplicity of illustration, we take very simple linear combinations involving small integers. These happen to suffice, but in general one would use a random number generator and possibly hundreds of digits to better approximate the probability-one chance of success that is implied in a continuum model of the coefficient space.

4 Computing Critical Points of Projections

An application of Corollary 1 is to compute the critical points of an irreducible curve $X \subset \mathbb{C}^N$ with respect to a nonconstant linear projection $\pi : X \to \mathbb{C}$. In particular, assume that $f = \{f_1, \ldots, f_{N-1}\}$ is a polynomial system on $\mathbb{C}^N$ such that X is an irreducible component of $\mathcal{V}(f)$ which has multiplicity one with respect to f. A critical point of π with respect to X is a point $x \in X$ such that either

- x is a smooth point and $d\pi$ is zero on the tangent space of X at x; or
- x is a singular point of X.

In terms of rank-deficiency sets, the set of critical points is the set of points on X such that

$$\operatorname{rank} \begin{bmatrix} d\pi \\ df_1 \\ \vdots \\ df_{N-1} \end{bmatrix} \leq N - 1. \tag{4}$$

With this setup, there are finitely many critical points. In [7], which includes an implementation of the curve decomposition algorithm of [18], a finite superset of the critical points are needed to compute a cellular decomposition of the real points of X. In fact, the points that are not critical points simply make the cellular decomposition finer which can be merged away in a post-processing step. Hence, one needs to compute at least one point in each connected component in $X \times \mathbb{P}^{N-1}$ intersected with the solution set in $\mathbb{C}^N \times \mathbb{P}^{N-1}$ of

$$\begin{bmatrix} f_1 \\ \vdots \\ f_{N-1} \\ \begin{bmatrix} d\pi \\ df_1 \\ \vdots \\ df_{N-1} \end{bmatrix} \cdot \xi \end{bmatrix} = 0.$$

The advantage here is that we obtain a finite superset of the critical points using one homotopy regardless of the possibly different dimensions of the corresponding null spaces, i.e., there is no need to cascade down the possible null space dimensions.

The setup above naturally extends to computing witness point supersets for the critical set of dimension $k-1$ of an irreducible component of dimension k, e.g., critical curves of a surface.

5 Example

Consider the 12-bar spherical linkage from [27,28]. This device can be viewed as 20 rigid rods meeting in spherical joints at 9 points, or since a loop of three

such rods forms a rigid triangle, as 12 rigid links meeting in rotational hinges with the axes of rotation all intersecting at a central point. The arrangement is most clearly seen in Fig. 2(c). The irreducible decomposition of the variety in $\mathbb{C}^{18}$ for the polynomial system F defined below for this linkage was first computed in [11] and summarized in Table 1. Here, we consider computing a superset of the critical points of the the curve C which is the union of the eight one-dimensional irreducible components having degree 36 with respect to the projection π defined below in (5). We will compare approaches computed using `Bertini` [4].

Table 1. Decomposition of 12-bar spherical linkage system.

Dimension	Degree	# components
3	8	2
2	4	2
	8	14
	12	12
	16	1
	20	4
	24	1
1	4	6
	6	2

The ground link for the linkage is specified by fixing three points, namely $P_0 = (0, 0, 0)$, $P_7 = (-1, 1, -1)$, and $P_8 = (-1, -1, -1)$. The three coordinates of the other six points, $P_1, \ldots, P_6$, are the 18 variables of polynomial system $F : \mathbb{C}^{18} \to \mathbb{C}^{17}$. The 17 polynomials in F are the following quadratics:

$$G_{ij} = \|P_i - P_j\|^2 - 4,$$
$$(i, j) \in \{(1,2), (3,4), (5,6), (1,5), (2,6), (3,7), (4,8), (1,3), (2,4), (5,7), (6,8)\};$$
$$H_k = \|P_k\|^2 - 3,$$
$$k \in \{1, 2, 3, 4, 5, 6\}.$$

Denoting the coordinates of P_i as P_{i1}, P_{i2}, P_{i3}, we choose[2] a projection map $\pi : \mathbb{C}^{18} \to \mathbb{C}$ defined by

$$\pi(P) = \tfrac{3}{5}P_{11} + \tfrac{13}{17}P_{12} - \tfrac{5}{16}P_{13} + \tfrac{26}{27}P_{21} - \tfrac{1}{10}P_{22} + \tfrac{1}{6}P_{23} + \tfrac{3}{5}P_{31} + \tfrac{7}{17}P_{32} + \tfrac{3}{10}P_{33} + \tfrac{1}{4}P_{41} - \tfrac{4}{5}P_{42} + \tfrac{1}{3}P_{43} + \tfrac{18}{25}P_{51} + \tfrac{14}{29}P_{52} - \tfrac{12}{13}P_{53} - \tfrac{17}{30}P_{61} - \tfrac{5}{17}P_{62} + \tfrac{13}{20}P_{63} \quad (5)$$

and consider the following system defined on $C \times \mathbb{P}^{17} \subset \mathbb{C}^{18} \times \mathbb{P}^{17}$:

[2] As before, we choose simple rational coefficients for simplicity of presentation.

$$f(P,\xi) = \begin{bmatrix} F(P) \\ \begin{bmatrix} d\pi \\ dF(P) \end{bmatrix} \cdot \xi \end{bmatrix}.$$

Since each irreducible component in C has multiplicity one with respect to F, the irreducible components of $\mathcal{V}(f) \cap (C \times \mathbb{P}^{17})$ must be of the form $\{x\} \times L$ for some point $x \in C$ and linear space $L \subset \mathbb{P}^{17}$. We aim to compute all such points x.

With traditional methods, one would need to consider various dimensions of the corresponding null spaces L. The advantage is that one obtains additional information, namely witness point supersets for the irreducible components. The first approach is to consider each possible dimension of $\mathbb{P}^{17}$ independently. Since the zero-dimensional case is equivalent in terms of the setup and number of paths to the new approach discussed below, we will just quickly summarize what would be needed to perform this full computation. In particular, for each $0 \leq i \leq 16$, starting with a witness set for $C \times \mathbb{P}^{17}$, the corresponding start system, after possible randomization, would require tracking $36 \cdot (17 - i)$, totaling 5508, paths related to moving linear slices and the same number of paths to compute witness point supersets.

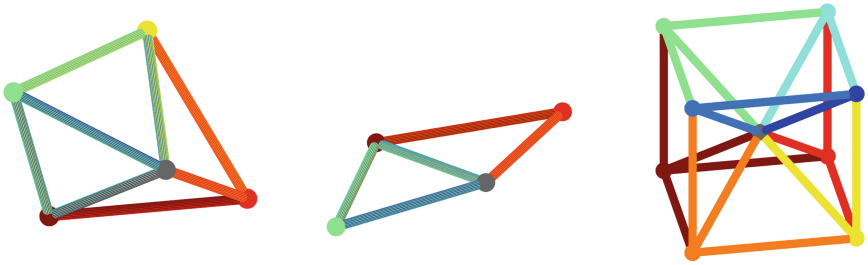
Rather than treat each dimension independently, another option is to cascade down through the dimensions, e.g., using the regenerative extension [15]. The implementation in `Bertini`, starting with a witness set for $C \times \mathbb{P}^{17}$, requires tracking 6276 paths for solving as well as tracking 3216 paths related to moving linear slices. Using 64, 2.3 GHz processors, this computation took 618 s.

Instead of using a method designed for computing witness point supersets, our new approach uses one homotopy to compute a point on each connected component. This is all that is needed for the current application via Corollary 1. Since $d\pi$ is constant and dF is a matrix with linear entries, we take our start system to be

$$g(P,\xi) = \begin{bmatrix} F(P) \\ \xi_0 \\ \ell_1(P) \cdot \xi_1 \\ \vdots \\ \ell_{17}(P) \cdot \xi_{17} \end{bmatrix}$$

restricted to $C \times \mathbb{P}^{17}$ where each ℓ_i is a random linear polynomial. In particular, $\mathcal{V}(g) \cap (C \times \mathbb{P}^{17})$ consists of $d = 36 \cdot \binom{17}{1} = 612$ points, each of which is nonsingular with respect to g. The 612 solutions can be computed from a witness set for C by tracking 612 paths related to moving linear slices. Then, a point on each connected component of $\mathcal{V}(f) \cap (C \times \mathbb{P}^{17})$ is computed via Corollary 1 by tracking 612 paths. This computation in total, using the same parallel setup as above, took 20 s.

Of the 612 paths, 492 diverge to infinity while 120 have finite endpoints. Of the 120 finite endpoints of the form (P, ξ), 78 are real (i.e., have $P \in \mathbb{R}^{18}$) with 22 distinct real points P since some points appear with multiplicity while others have a null space with dimension greater than one so that the same P can appear



(a) (b) (c)

Fig. 2. Solutions to the 12 bar spherical linkage obtained from the critical point computation: (a) an equilateral spherical four-bar configuration, corresponding to a non-singular critical point on a degree four irreducible component; (b) a degenerate configuration, coming from the intersection of such a component with a higher-dimensional irreducible component; (c) a rigid configuration arising from the intersection of the irreducible curves of degree six.

with several different null directions ξ. In detail, the breakdown of the 22 real points is as follows:

- 14 real points are the endpoint of one path each. These points are smooth points of C with rank$dF = 17$. Each lie on one of the degree 4 irreducible components of C and is an equilateral spherical four-bar linkage of the type illustrated in Fig. 2(a).
- 6 real points are the endpoint of 10 paths each. Each of these points has rank$dF = 12$ with rank $\begin{bmatrix} d\pi \\ dF \end{bmatrix} = 13$ and arise where an irreducible component of degree 4 in C intersects another irreducible component of $\mathcal{V}(F)$. The corresponding 12-bar linkage appears as in Fig. 2(b).
- 2 real points are the endpoint of 2 paths each. Each of these points P has rank$dF = 16$ and rank $\begin{bmatrix} d\pi \\ dF \end{bmatrix} = 17$ so that the corresponding null vector $\xi \in \mathbb{P}^{17}$ is unique. Hence, the points (P, ξ) have multiplicity 2 with respect to f. These points correspond to a rigid arrangement as shown in Fig. 2(c), one the mirror image of the other.

To clarify the accounting, note that $14 \cdot 1 + 6 \cdot 10 + 2 \cdot 2 = 78$.

6 Conclusion

We have described an algorithmic approach for constructing one homotopy that yields a finite superset of solutions to a polynomial system containing at least one point on each connected component of the solution set. This idea naturally leads to homotopies for solving elimination problems, such as computing critical points of projections as well as other rank-constraint problems. This method

allows one to compute such points directly without having to cascade through all the possible dimensions of the auxiliary variables. This can provide considerable computational savings, as we have demonstrated on an example arising in kinematics, where the endpoints of a single homotopy include all the critical points on a curve even though the associated null spaces at these points have various dimensions.

We note that our approach has application to numerical elimination theory but in that case leaves an open problem concerning sorting isolated from non-isolated points. In the classical setting, when one finds a superset of the isolated solutions, one can sift out the set of isolated solutions from a superset by using, for example, either the global homotopy membership test [23] or the numerical local dimension test [3]. In the elimination setting, a modified version of the homotopy membership test as developed in [12] can sort out which points are isolated under projection, but there is no local dimension test in this setting as yet.

References

1. Aubry, P., Rouillier, F., Safey El Din, M.: Real solving for positive dimensional systems. J. Symbolic Comput. **34**(6), 543–560 (2002)
2. Bates, D.J., Hauenstein, J.D., Peterson, C., Sommese, A.J.: Numerical decomposition of the rank-deficiency set of a matrix of multivariate polynomials. In: Robbiano, L., Abbott, J. (eds.) Approximate Commutative Algebra. Texts and Monographs in Symbolic Computation (A Series of the Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria), pp. 55–77. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-211-99314-9_2
3. Bates, D.J., Hauenstein, J.D., Peterson, C., Sommese, A.J.: A numerical local dimensions test for points on the solution set of a system of polynomial equations. SIAM J. Numer. Anal. **47**(5), 3608–3623 (2009)
4. Bates, D.J., Hauenstein, J.D., Sommese, A.J., Wampler, C.W.: Bertini: Software for numerical algebraic geometry. bertini.nd.edu
5. Bates, D.J., Hauenstein, J.D., Sommese, A.J., Wampler, C.W.: Numerically Solving Polynomial Systems with Bertini. SIAM, Philadelphia (2013)
6. Brake, D.A., Bates, D.J., Hao, W., Hauenstein, J.D., Sommese, A.J., Wampler, C.W.: Bertini_real: Software for one- and two-dimensional real algebraic sets. In: Hong, H., Yap, C. (eds.) ICMS 2014. LNCS, vol. 8592, pp. 175–182. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44199-2_29
7. Brake, D.A., Bates, D.J., Hao, W., Hauenstein, J.D., Sommese, A.J., Wampler, C.W.: Bertini_real: Numerical decomposition of real algebraic curves and surfaces. ACM Trans. Math. Softw. **44**(1), 10 (2017)
8. Chern, S.: Characteristic classes of hermitian manifolds. Annals Math. **47**(1), 85–121 (1946)
9. Draisma, J., Horobet, E., Ottaviani, G., Sturmfels, B., Thomas, R.R.: The Euclidean distance degree of an algebraic variety. Found. Comput. Math. **16**(1), 99–149 (2016)
10. Fulton, W.: Intersection Theory. Springer, Heidelberg (1998). https://doi.org/10.1007/978-1-4612-1700-8

11. Hauenstein, J.D.: Numerically computing real points on algebraic sets. Acta Appl. Math. **125**(1), 105–119 (2013)
12. Hauenstein, J.D., Sommese, A.J.: Membership tests for images of algebraic sets by linear projections. Appl. Math. Comput. **219**(12), 6809–6818 (2013)
13. Hauenstein, J.D., Sommese, A.J., Wampler, C.W.: Regenerative cascade homotopies for solving polynomial systems. Appl. Math. Comput. **218**(4), 1240–1246 (2011)
14. Hauenstein, J.D., Wampler, C.W.: Isosingular sets and deflation. Found. Comp. Math. **13**(3), 371–403 (2013)
15. Hauenstein, J.D., Wampler, C.W.: Unification and extension of intersection algorithms in numerical algebraic geometry. Appl. Math. Comput. **293**, 226–243 (2017)
16. Huber, B., Sturmfels, B.: A polyhedral method for solving sparse polynomial systems. Math. Comp. **64**(212), 1541–1555 (1995)
17. Leykin, A.: Numerical primary decomposition. In: ISSAC 2008, pp. 165–172. ACM, New York (2008)
18. Lu, Y., Bates, D.J., Sommese, A.J., Wampler, C.W.: Finding all real points of a complex curve. Contemp. Math. **448**, 183–205 (2007)
19. Morgan, A.P., Sommese, A.J.: Coefficient-parameter polynomial continuation. Appl. Math. Comput. 29(2), 123–160 (1989). Errata: Appl. Math. Comput. **51**(207) (1992)
20. Morgan, A.P., Sommese, A.J.: A homotopy for solving general polynomial systems that respects m-homogeneous structures. Appl. Math. Comput. **24**, 101–113 (1987)
21. Rouillier, F., Roy, M.-F., Safey El Din, M.: Finding at least one point in each connected component of a real algebraic set defined by a single equation. J. Complex. **16**(4), 716–750 (2000)
22. Sommese, A.J., Verschelde, J.: Numerical homotopies to compute generic points on positive dimensional algebraic sets. J. Complex. **16**(3), 572–602 (2000)
23. Sommese, A.J., Verschelde, J., Wampler, C.W.: Numerical irreducible decomposition using projections from points on the components. Contemp. Math. **286**, 37–51 (2001)
24. Sommese, A.J., Wampler, C.W.: Numerical algebraic geometry. In: The Mathematics of Numerical Analysis (Park City, UT, 1995). Lectures in Applied Mathematics, vol. 32, pp. 749–763. AMS, Providence, RI (1996)
25. Sommese, A.J., Wampler, C.W.: The Numerical Solution of Systems of Polynomials Arising in Engineering and Science. World Scientific, Singapore (2005)
26. Verschelde, J., Cools, R.: Symbolic homotopy construction. Appl. Algebra Eng. Commun. Comput. **4**(3), 169–183 (1993)
27. Wampler, C.W., Larson, B., Erdman, A.: A new mobility formula for spatial mechanisms. In: Proceedings of the DETC/Mechanisms and Robotics Conference. ASME (2007). paper DETC2007-35574
28. Wampler, C.W., Hauenstein, J.D., Sommese, A.J.: Mechanism mobility and a local dimension test. Mech. Mach. Theory **46**(9), 1193–1206 (2011)
29. Wang, D.: Elimination Methods. Springer, Heidelberg (2001). https://doi.org/10.1007/978-3-7091-6202-6
30. Wu, W., Reid, G.: Finding points on real solution components and applications to differential polynomial systems. In: ISSAC 2013, pp. 339–346. ACM, New York (2013)

Implementing Fast Carryless Multiplication

Joris van der Hoeven(✉), Robin Larrieu, and Grégoire Lecerf

Laboratoire d'informatique de l'École polytechnique LIX, UMR 7161 CNRS,
Campus de l'École polytechnique 1, rue Honoré d'Estienne d'Orves
Bâtiment Alan Turing, CS35003, 91120 Palaiseau, France
{vdhoeven,larrieu,lecerf}@lix.polytechnique.fr

Abstract. The efficient multiplication of polynomials over the finite field $\mathbb{F}_2$ is a fundamental problem in computer science with several applications to geometric error correcting codes and algebraic crypto-systems. In this paper we report on a new algorithm that leads to a practical speed-up of about two over previously available implementations. Our current implementation assumes a modern AVX2 and CLMUL enabled processor.

1 Introduction

Modern algorithms for fast polynomial multiplication are generally based on *evaluation-interpolation* strategies and more particularly on the *discrete Fourier transform* (DFT). Taking coefficients in the finite field $\mathbb{F}_2$ with two elements, the problem of multiplying in $\mathbb{F}_2[x]$ is also known as *carryless integer multiplication* (assuming binary notation). The aim of this paper is to present a practically efficient solution for large degrees.

One major obstruction to evaluation-interpolation strategies over small finite fields is the potential lack of evaluation points. The customary remedy is to work in suitable extension fields. Remains the question of how to reduce the incurred overhead as much as possible.

More specifically, it was shown in [7] that multiplication in $\mathbb{F}_2[x]$ can be done efficiently by reducing it to polynomial multiplication over the *Babylonian field* $\mathbb{F}_{2^{60}}$. Part of this reduction relied on Kronecker segmentation, which involves an overhead of a factor two. In this paper, we present a variant of a new algorithm from [11] that removes this overhead almost entirely. We also report on our MATHEMAGIX implementation that is roughly twice as efficient as before.

1.1 Related Work

For a long time, the best known algorithm for carryless integer multiplication was Schönhage's triadic variant [16] of Schönhage–Strassen's algorithm [17] for integer multiplication: it achieves a complexity $O(n \log n \log\log n)$ for the multiplication of two polynomials of degree n. Recently [8], Harvey, van der Hoeven and Lecerf proved the sharper bound $O(n \log n 8^{\log^* n})$, but also showed that several of the new ideas could be used for faster practical implementations [7].

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 121–136, 2017.
https://doi.org/10.1007/978-3-319-72453-9_9

More specifically, they showed how to reduce multiplication in $\mathbb{F}_2[x]$ to DFTs over $\mathbb{F}_{2^{60}}$, which can be computed efficiently due to the existence of many small prime divisors of $2^{60}-1$. Their reduction relies on *Kronecker segmentation*: given two input polynomials $A(x) = \sum_{0 \leqslant i < n} a_i x^i$ and $B(x) = \sum_{0 \leqslant i < n} a_i x^i$ in $\mathbb{F}_2[x]$, one cuts them into chunks of 30 bits and forms $\tilde{A}(y,z) = \sum_{i=0}^{m-1} \sum_{j=0}^{29} a_{30i+j} z^j y^i$ and $\tilde{B}(y,z) = \sum_{i=0}^{m-1} \sum_{j=0}^{29} b_{30i+j} z^j y^i$, where $m = \lceil n/30 \rceil$ (the least integer $\geqslant n/30$). Hence $A(x) = \tilde{A}(x^{30}, x)$, $B(x) = \tilde{B}(x^{30}, x)$, and the product $C = AB$ satisfies $C(x) = \tilde{C}(x^{30}, x)$, where $\tilde{C} = \tilde{A}\tilde{B}$. Now $\tilde{A}$ and $\tilde{B}$ are multiplied in $\mathbb{F}_{2^{60}}[x]$ by reinterpreting z as the generator of $\mathbb{F}_{2^{60}}$. The recovery of $\tilde{C}$ is possible since its degree in z is bounded by $2 \cdot 29 = 58 < 60$. However, in terms of input size, half of 60 coefficients of $\tilde{A}(y,z)$ and $\tilde{B}(y,z)$ in z are "left blank", when reinterpreted inside $\mathbb{F}_{2^{60}}$. Consequently, this reduction method based on Kronecker segmentation involves a constant overhead of roughly 2. In fact, when considering algorithms with asymptotically softly linear costs, comparing relative input sizes gives a rough approximation of the relative costs.

Recently van der Hoeven and Larrieu [11] have proposed a new way to reduce multiplication of polynomials in $\mathbb{F}_q[x]$ to the computation of DFTs over an extension $\mathbb{F}_{q^\ell}$. Roughly speaking, they have shown that the DFT of a polynomial in $\mathbb{F}_{q^\ell}[x]$ could be computed almost ℓ times faster if its coefficients happen to lie in the subfield $\mathbb{F}_q$. Using their algorithm, called the *Frobenius FFT*, it is theoretically possible to avoid the overhead of Kronecker segmentation, and thereby to gain a factor of two with respect to [7]. However, application of the Frobenius FFT as described in [11] involves computations in all intermediate fields $\mathbb{F}_{q^e}$ between $\mathbb{F}_q$ and $\mathbb{F}_{q^\ell}$. This makes the theoretical speed-up of two harder to achieve and practical implementations more cumbersome.

Besides Schönhage–Strassen type algorithms, let us mention that other strategies such as the *additive Fourier transform* have been developed for $\mathbb{F}_{2^k}[x]$ [4,15]. A competitive implementation based on the latter transform has been achieved very recently by Chen et al. [2]—notice that their preprint [2] does not take into account our new implementation. For more historical details on the complexity of polynomial multiplication we refer the reader to the introductions of [7,8] and to the book by von zur Gathen and Gerhard [5].

1.2 Results and Outline of the Paper

This paper contains two main results. In Sect. 3, we describe a variant of the Frobenius DFT for the special extension of $\mathbb{F}_{2^{60}}$ over $\mathbb{F}_2$. Using a single rewriting step, this new algorithm reduces the computation of a Frobenius DFT to the computation of an ordinary DFT over $\mathbb{F}_{2^{60}}$, thereby avoiding computations in any intermediate fields $\mathbb{F}_{2^e}$ with $1 < e < 60$ and $e \mid 60$.

Our second main result is a practical implementation of the new algorithm and our ability to indeed gain a factor that approaches two with respect to our previous work. We underline that in both cases, DFTs over $\mathbb{F}_{2^{60}}$ represent the bulk of the computation, but the lengths of the DFTs are halved for the new

algorithm. Hence, the observed acceleration is due to our new algorithm and not the result of *ad hoc* code tuning or hardware specific optimizations.

In Sect. 4, we present some of the low level implementation details concerning the new rewriting step. Our timings are presented in Sect. 5. Our implementation outperforms the reference library GF2X version 1.2 developed by Brent et al. [1] for multiplying polynomials in $\mathbb{F}_2[x]$. We also outperform the recent implementation by Chen et al. [2]. Finally, the evaluation-interpolation strategy used by our algorithm is particularly well suited for multiplying matrices of polynomials over $\mathbb{F}_2$, as reported in Sect. 5.

2 Prerequisites

Discrete Fourier Transforms. Let ω be a primitive root of unity of order n in $\mathbb{F}_q$. The *discrete Fourier transform* (DFT) of an n-tuple $a = (a_0, \ldots, a_{n-1}) \in \mathbb{F}_q^n$ with respect to ω is $\mathrm{DFT}_\omega(a) := (\hat{a}_0, \ldots, \hat{a}_{n-1}) \in \mathbb{F}_q^n$, where

$$\hat{a}_i := a_0 + a_1\omega^i + \cdots + a_{n-1}\omega^{(n-1)i}.$$

Hence $\hat{a}_i$ is the evaluation of the polynomial $A(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ at ω^i. For simplicity we often identify A with a and we simply write $\mathrm{DFT}_\omega(A)$. The inverse transform is related to the direct transform via $\mathrm{DFT}_\omega^{-1} = n^{-1}\,\mathrm{DFT}_{\omega^{-1}}$, which follows from the well known formula

$$\mathrm{DFT}_{\omega^{-1}}(\mathrm{DFT}_\omega(a)) = na.$$

If n properly factors as $n = n_1n_2$, then ω^{n_1} is an n_2-th primitive root of unity and ω^{n_2} is an n_1-th primitive root of unity. Moreover, for any $i_1 \in \{0, \ldots, n_1 - 1\}$ and $i_2 \in \{0, \ldots, n_2 - 1\}$, we have

$$\begin{aligned}\hat{a}_{i_1n_2+i_2} &= \sum_{0 \leqslant k_1 < n_1} \sum_{0 \leqslant k_2 < n_2} a_{k_2n_1+k_1}\omega^{(k_2n_1+k_1)(i_1n_2+i_2)} \\ &= \sum_{0 \leqslant k_1 < n_1} \omega^{k_1i_2} \left(\sum_{0 \leqslant k_2 < n_2} a_{k_2n_1+k_1}(\omega^{n_1})^{k_2i_2} \right) (\omega^{n_2})^{k_1i_1}. \qquad (1)\end{aligned}$$

If $\mathcal{A}_1$ and $\mathcal{A}_2$ are algorithms for computing DFTs of length n_1 and n_2, we may use (1) to construct an algorithm for computing DFTs of length n as follows. For each $k_1 \in \{0, \ldots, n_1 - 1\}$, the sum inside the brackets corresponds to the i_2-th coefficient of a DFT of the n_2-tuple $(a_{0n_1+k_1}, \ldots, a_{(n_2-1)n_1+k_1}) \in \mathbb{F}_q^{n_2}$ with respect to ω^{n_1}. Evaluating these *inner DFTs* requires n_1 calls to $\mathcal{A}_2$. Next, we multiply by the *twiddle factors* $\omega^{k_1i_2}$, at a cost of n operations in $\mathbb{F}_q$. Finally, for each $i_2 \in \{0, \ldots, n_2 - 1\}$, the outer sum corresponds to the i_1-th coefficient of a DFT of an n_1-tuple in $\mathbb{F}_q^{n_1}$ with respect to ω^{n_2}. These *outer DFTs* require n_2 calls to $\mathcal{A}_1$. Iterating this decomposition for further factorizations of n_1 and n_2 yields the seminal Cooley–Tukey algorithm [3].

Frobenius Fourier Transforms. Let A be a polynomial in $\mathbb{F}_q[x]$ and let ω be a primitive root of unity in some extension $\mathbb{F}_{q^\ell}$ of $\mathbb{F}_q$. We write ϕ_q for the Frobenius map $a \mapsto a^q$ in $\mathbb{F}_{q^\ell}$ and notice that

$$A(\phi_q(a)) = \phi_q(A(a)), \tag{2}$$

for any $a \in \mathbb{F}_{q^\ell}$. This formula implies many nontrivial relations for the DFT of A: if $\omega^i = \phi_q^{\circ k}(\omega^j)$, then we have $A(\omega^i) = \phi_q^{\circ k}(A(\omega^j))$. In other words, some values of the DFT of A can be deduced from others, and the advantage of the Frobenius transform introduced in [11] is to restrict the bulk of the evaluations to a minimum number of points.

Let n denote the order of the root ω, and consider the set $\Omega = \{1, \omega, \omega^2, \ldots, \omega^{n-1}\}$. This set is clearly globally stable under ϕ_q, so the group $\langle \phi_q \rangle$ generated by ϕ_q acts naturally on it. This action partitions Ω into disjoint orbits. Assume that we have a section Σ of Ω that contains exactly one element in each orbit. Then formula (2) allows us to recover $\mathrm{DFT}_\omega(A)$ from the evaluations of A at each of the points in Σ. The vector $(A(\sigma))_{\sigma \in \Sigma}$ is called the *Frobenius DFT* of A.

3 Fast Reduction from $\mathbb{F}_2[x]$ to $\mathbb{F}_{2^{60}}[x]$

3.1 Variant of the Frobenius DFT

To efficiently reduce a multiplication in $\mathbb{F}_2[x]$ into DFTs over $\mathbb{F}_{2^{60}}$, we use an order n that divides $2^{60} - 1$ and such that $n = 61m$ for some integer m. We perform the decomposition (1) with $n_1 = m$ and $n_2 = 61$. Let ω be a primitive n-th root of unity in $\mathbb{F}_{2^{60}}$. The discrete Fourier transform of $A \in \mathbb{F}_2[x]_{<n}$, given by $(A(1), A(\omega), A(\omega^2), \ldots, A(\omega^{n-1})) \in \mathbb{F}_{2^{60}}^n$, can be reorganized into 61 slices as follows

$$\mathrm{DFT}_\omega(A) = ((A(\omega^{61i}))_{0 \leqslant i < m}, (A(\omega^{61i+1}))_{0 \leqslant i < m}, \ldots, (A(\omega^{61i+60}))_{0 \leqslant i < m}).$$

The variant of the Frobenius DFT of A that we introduce in the present paper corresponds to computing only the second slice:

$$\begin{aligned} E_\omega : \quad \mathbb{F}_2[x]_{<60m} &\to \mathbb{F}_{2^{60}}^m \\ A &\mapsto (A(\omega^{61i+1}))_{0 \leqslant i < m}. \end{aligned}$$

Let us show that this transform is actually a bijection. The following lemma shows that the slices $(A(\omega^{61i+2}))_{0 \leqslant i < m}, \ldots, (A(\omega^{61i+60}))_{0 \leqslant i < m}$ can be deduced from the second slice $(A(\omega^{61i+1}))_{0 \leqslant i < m}$ using the action of the Frobenius map ϕ_2.

Lemma 1. *Let $\Omega_i = \{\omega^{61j+i} : 0 \leqslant j < m\}$ for $1 \leqslant i < 61$. Then the action of $\langle \phi_2 \rangle$ is transitive on the pairwise disjoint sets $\Omega_1, \ldots, \Omega_{60}$.*

Proof. Let $1 \leqslant i < 61$ and $0 \leqslant j < m$, we have $\phi_2(\omega^{61j+i}) = \omega^{61j' + (2i \bmod 61)}$ for some integer $0 \leqslant j' < m$, so the action of $\langle \phi_2 \rangle$ onto $\Omega_1, \ldots, \Omega_{60}$ is well defined.

Notice that 2 is primitive for the multiplicative group $\mathbb{F}_{61}^{\times}$. This implies that for any $1 \leqslant i < 61$ there exists k such that $2^k = i \bmod 61$. Consequently we have $\phi_2^{\circ k}(\omega^{61j+1}) = \omega^{61j'+i}$ for some $0 \leqslant j' < m$, whence $\phi_2^{\circ k}(\Omega_1) \subseteq \Omega_i$. Since ϕ_2 is injective the latter inclusion is an equality.

If we were needed the complete $\mathrm{DFT}_\omega(A)$, then we would still have to compute the first slice $(A(\omega^{61i}))_{0 \leqslant i < m}$. The second main new idea with respect to [11] is to discard this first slice and to restrict ourselves to input polynomials A of degrees $<60m$. In this way, E_ω can be inverted, as proved in the following proposition.

Proposition 1. *E_ω is bijective.*

Proof. The dimensions of the source and destination spaces of E_ω over $\mathbb{F}_2$ being the same, it suffices to prove that E_ω is injective. Let $A \in \mathbb{F}_2[x]_{<60m}$ be such that $E_\omega(A) = 0$. By construction, A vanishes at m distinct values, namely ω^{61i+1} for $0 \leqslant i < m$. Under the action of $\langle \phi_2 \rangle$ it also vanishes at $60(m-1)$ other values by Lemma 1, whence $A = 0$.

Remark 1. The transformation E_ω being bijective is due to the fact that 2 is primitive in the multiplicative group $\mathbb{F}_{61}^{\times}$. Among the prime divisors of $2^{60} - 1$, the factors 3, 5, 11 and 13 also have this property, but taking $n_2 = 61$ allows us to divide the size of the evaluation-interpolation scheme by 60, which is optimal.

3.2 Frobenius Encoding

We decompose the computation of E_ω into two routines. The first routine is written F_ω and called the *Frobenius encoding*:

$$\begin{aligned} F_\omega : \quad & \mathbb{F}_2[x]_{<60m} \to \mathbb{F}_{2^{60}}[x]_{<m} \\ A = \sum_{0 \leqslant k < 60m} a_k x^k \mapsto & \sum_{0 \leqslant k < m} \omega^k \left(\sum_{0 \leqslant l < 60} a_{k+ml} \theta^l \right) x^k, \text{where } \theta = \omega^m. \end{aligned} \tag{3}$$

Below, we will choose θ in such a way that F_ω is essentially a simple reorganization of the coefficients of A.

We observe that the coefficients of $F_\omega(A)$ are part of the values of the inner DFTs of A in the Cooley–Tukey formula (1), applied with $n_1 = m$ and $n_2 = 61$. The second task is the computation of the corresponding outer DFT of order m:

$$\begin{aligned} \mathrm{DFT}_{\tilde{\omega}} : \quad & \mathbb{F}_{2^{60}}[x]_{<m} \to \mathbb{F}_{2^{60}}^m \\ & \tilde{A} \mapsto (\tilde{A}(\tilde{\omega}^i))_{0 \leqslant i < m}, \text{ where } \tilde{\omega} = \omega^{61}. \end{aligned}$$

Proposition 2. $E_\omega = \mathrm{DFT}_{\tilde{\omega}} \circ F_\omega$.

Proof. This formula follows from (1):

$$A(\omega^{61i+1}) = \sum_{0 \leqslant k < m} \omega^k \left(\sum_{0 \leqslant l < 61} a_{k+ml} \theta^l \right) \tilde{\omega}^{ki} = F_\omega(A)(\tilde{\omega}^i).$$

Summarizing, we have reduced the computation of a DFT of size $60n/61$ over $\mathbb{F}_2$ to a DFT of size $m = n/61$ over $\mathbb{F}_{2^{60}}$. This reduction preserves data size.

3.3 Direct Transforms

The computation of F_ω involves the evaluation of m polynomials in $\mathbb{F}_2[x]_{<60}$ at $\theta = \omega^m \in \mathbb{F}_{2^{60}}$. In order to perform these evaluations fast, we fix the representation of $\mathbb{F}_{2^{60}} = \mathbb{F}_2[z]/(\mu(z))$ and the primitive root ν of unity of maximal order $2^{60} - 1$ to be given by

$$\begin{aligned} \mu(z) &= (z^{61} - 1)/(z - 1) \\ \nu &= z^{18} + z^6 + 1 \bmod \mu(z). \end{aligned}$$

Setting $\omega = \nu^{(2^{60}-1)/n}$ and $\theta = \nu^{(2^{60}-1)/61}$, it can be checked that $\theta = z \bmod \mu(z)$. Evaluation of a polynomial in $\mathbb{F}_2[x]_{<60}$ at θ can now be done efficiently.

Algorithm 1.
Input: $A(x) = \sum_{0 \leqslant i < 60m} a_i x^i$.
Output: $F_\omega(A)$.
Assumption: $n = 61m$ divides $2^{60} - 1$.

1. For $i = 0, \ldots, m-1$, build $P_i(z) = \sum_{0 \leqslant j < 60} a_{i+mj} z^j \bmod \mu(z) \in \mathbb{F}_{2^{60}}$.
2. Return $P_0 + \omega P_1 x + \omega^2 P_2 x^2 + \cdots + \omega^{m-1} P_{m-1} x^{m-1}$.

Proposition 3. *Algorithm 1 is correct.*

Proof. This deduces immediately from the definition of F_ω in formula (3), using the fact that $\theta = z \bmod \mu(z)$ in our representation.

Algorithm 2.
Input: $A \in \mathbb{F}_2[x]_{<60m}$.
Output: $E_\omega(A)$.
Assumption: $n = 61m$ divides $2^{60} - 1$.

1. Compute the Frobenius encoding $\tilde{A}(x) \in \mathbb{F}_{2^{60}}[x]_{<m}$ of A by Algorithm 1.
2. Compute the DFT of $\tilde{A}$ with respect to $\tilde{\omega}$.

Proposition 4. *Algorithm 2 is correct.*

Proof. The correctness simply follows from Propositions 2 and 3.

3.4 Inverse Transforms

By combining Propositions 1 and 2, the map F_ω is invertible and its inverse may be computed by the following algorithm.

Algorithm 3.
Input: $\tilde{A}(x) = \sum_{i \geqslant 0} \tilde{a}_i x^i \in \mathbb{F}_{2^{60}}[x]_{<m}$.
Output: $F_\omega^{-1}(\tilde{A})$.
Assumption: $n = 61m$ divides $2^{60} - 1$.

1. For $i = 0, \ldots, m-1$, build the preimage $P_i(z) := \sum_{0 \leqslant j < 60} p_{i,j} z^j$ of $\omega^{-i} \tilde{a}_i$.
2. Return $\sum_{0 \leqslant i < m} \sum_{0 \leqslant j < 60} p_{i,j} x^{i+mj}$.

Proposition 5. *Algorithm 3 is correct.*

Proof. This is a straightforward inversion of Algorithm 1 .

Algorithm 4.
Input: $\hat{a} \in \mathbb{F}_{2^{60}}^m$.
Output: $E_\omega^{-1}(\hat{a})$.
Assumption: $n = 61m$ divides $2^{60} - 1$.

1. Compute the inverse DFT $\tilde{A} \in \mathbb{F}_{2^{60}}[x]_{<m}$ of $\hat{a}$ with respect to $\tilde{\omega}$.
2. Compute the Frobenius decoding A of $\tilde{A}$ by Algorithm 3 and return A.

Proposition 6. *Algorithm 4 is correct.*

Proof. The correctness simply follows from Propositions 2 and 5.

3.5 Multiplication in $\mathbb{F}_2[x]$

Using the standard technique of multiplication by evaluation-interpolation, we may now compute products in $\mathbb{F}_2[x]$ as follows:

Algorithm 5.
Input: $A, B \in \mathbb{F}_2[x]_{<\ell}$.
Output: AB

1. Let $m \geqslant (2\ell - 1)/60$ be such that $n = 61m$ divides $2^{60} - 1$.
2. Let $\omega = \nu^{(2^{60}-1)/n}$ be the privileged root of unity of order n.
3. Compute $E_\omega(A)$ and $E_\omega(B)$ by Algorithm 2.
4. Compute $\hat{c}$ as the entry-wise product of $E_\omega(A)$ and $E_\omega(B)$.
5. Compute $C(x) = E_\omega^{-1}(\hat{c})$ by Algorithm 4 and return C.

Proposition 7. *Algorithm 4 is correct.*

Proof. The correctness simply follows from Propositions 4 and 6 and using the fact that $E_\omega(AB) = E_\omega(A)E_\omega(B)$, since $m \geqslant (2\ell - 1)/60$.

For step 1, the actual determination of m has been discussed in [7, Sect. 3]. In fact it is often better not to pick the smallest possible value for m but a slightly larger one that is also very smooth. Since $2^{60} - 1$ admits many small prime divisors, such smooth values of m usually indeed exist.

4 Implementation Details

We follow Intel's terminology and use the term *quad word* to denote a unit of 64 bits of data. In the rest of the paper we use the C99 standard for presenting our source code. In particular a quad word representing an unsigned integer is considered of type `uint64_t`.

Our implementations are done for an AVX2-enabled processor and an operating system compliant to System V Application Binary Interface. The C++ library numerix of Mathemagix [13] (http://www.mathemagix.org) defines wrappers for AVX types. In particular, `avx_uint64_t` represents an SIMD vector of 4 elements of type `uint64_t`. Recall that the platform disposes of 16 AVX registers which must be allocated accurately in order to minimize read and write accesses to the memory.

Our new polynomial product is implemented in the justinline library of Mathemagix. The source code is freely available from revision 10681 of our SVN server (https://gforge.inria.fr/projects/mmx/). Main sources are in `justinline/src/frobenius_encode_f2_60.cpp` for the Frobenius encoding and in `justinline/mmx/polynomial_f2_amd64_avx2_clmul.mmx` for the top level functions. Related test and bench files are also available from dedicated directories of the justinline library. Let us further mention here that our Mathemagix functions may be easily exported to C++ [12].

4.1 Packed Representations

Polynomials over $\mathbb{F}_2$ are supposed to be given in *packed representation*, which means that coefficients are stored as a vector of contiguous bits in memory. For the implementation considered in this paper, a polynomial of degree $\ell-1$ is stored into $\lceil \ell/64 \rceil$ quad words, starting with the low-degree coefficients: the constant term is the least significant bit of the first word. The last word is suitably padded with zeros.

Reading or writing one coefficient or a range of coefficients of a polynomial in packed representation must be done carefully to avoid invalid memory access. Let A be such a polynomial of type `uint64_t*`. Reading the coefficient a_i of degree i in A is obtained as `(A[i >> 6] >> (i & 63)) & 1`. However, reading or writing a single coefficient should be avoided as much as possible for efficiency, so we prefer handling ranges of 256 bits. In the sequel the function of prototype

```
void load (avx_uint64_t& d, const uint64_t* A,
  const uint64_t& ℓ, const uint64_t& i, const uint64_t& e);
```

returns the $e \leqslant 256$ bits of A starting from i into d. Bits beyond position ℓ are considered to be zero.

For arithmetic operations in $\mathbb{F}_{2^{60}}$ we refer the reader to [7, Sect. 3.1]. In the sequel we only appeal to the function

```
uint64_t f2_60_mul (const uint64_t& a, const uint64_t& b);
```

that multiplies the two elements a and b of $\mathbb{F}_{2^{60}}$ in packed representation.

We also use a packed column-major representation for matrices over $\mathbb{F}_2$. For instance, an 8×8 bit matrix $(M_{i,j})_{0 \leqslant i < 8,\, 0 \leqslant j < 8}$ is encoded as a quad word whose $(8j+i)$-th bit is $M_{i,j}$. Similarly, a $256 \times \ell$ matrix $(M_{i,j})_{0 \leqslant i < 256,\, 0 \leqslant j < \ell}$ may be seen as a vector v of type `avx_uint64_t*`, so $M_{i,j}$ corresponds to the i-th bit of `v[`j`]`.

4.2 Matrix Transposition

The Frobenius encoding essentially boils down to matrix transpositions. Our main building block is 256×64 bit matrix transposition. We decompose this transposition in a suitable way with regards to data locality, register allocation and vectorization.

For the computation of general transpositions, we repeatedly make use of the well-known divide and conquer strategy: to transpose an $n \times \ell$ matrix M, where n and ℓ are even, we decompose $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$, where A, B, C, D are $n/2 \times \ell/2$ matrices; we swap the anti-diagonal blocks B and C and recursively transpose each block A, B, C, D.

Transposing Packed 8 × 8 Bit Matrices. The basic task we begin with is the transposition of a packed 8×8 bit matrix. The solution used here is borrowed from [18, Chap. 7, Sect. 3].

Function 6.
Input: $(M_{i,j})_{0 \leqslant i < 8,\, 0 \leqslant j < 8}$ in packed representation.
Output: The transpose $(N_{i,j})_{0 \leqslant i < 8,\, 0 \leqslant j < 8}$ of M in packed representation.

```
uint64_t
packed_matrix_bit_8x8_transpose (const uint64_t& M) {
 1. uint64_t N = M;
 2. static const uint64_t mask_4 = 0x00000000f0f0f0f0;
 3. static const uint64_t mask_2 = 0x0000cccc0000cccc;
 4. static const uint64_t mask_1 = 0x00aa00aa00aa00aa;
 5. uint64_t a;
 6. a = ((N >> 28) ^ N) & mask_4; N = N ^ a;
 7. a = a << 28; N = N ^ a;
 8. a = ((N >> 14) ^ N) & mask_2; N = N ^ a;
 9. a = a << 14; N = N ^ a;
10. a = ((N >> 7) ^ N) & mask_1;  N = N ^ a;
11. a = a << 7;  N = N ^ a;
12. return N; }
```

In steps 6 and 7, the anti-diagonal 4×4 blocks are swapped. In steps 8 and 9, the matrix N is seen as four 4×4 matrices whose anti-diagonal 2×2 blocks are swapped. In steps 10 and 11, the matrix N is seen as sixteen 2×2 matrices whose anti-diagonal elements are swapped. All in all, 18 instructions, 3 constants and one auxiliary variable are needed to transpose a packed 8×8 bit matrix in this way.

One advantage of the above algorithm is that it admits a straightforward AVX vectorization that we will denote by

```
avx_uint64_t
avx_packed_matrix_bit_8x8_transpose (const avx_uint64_t& M);
```

This routine transposes four 8×8 bit matrices M_0, M_1, M_2, M_3 that are packed successively into an AVX register of type `avx_uint64_t`. We emphasize that this task is *not* the same as transposing a 32×8 or 8×32 bit matrices.

Remark 2. The BMI2 technology gives another method for transposing 8×8 bit matrices:

```
uint64_t mask = 0x0101010101010101;
uint64_t N= 0;
for (unsigned i = 0; i < 8; i++)
  N |= _pext_u64 (M, mask << i) << (8 * i);
```

The loop can be unrolled while precomputing the shift amounts and masks, which leads to a faster sequential implementation. Unfortunately this approach cannot be vectorized with the AVX2 technology. Other sequential solutions even exist, based on lookup tables or integer arithmetic, but their vectorization is again problematic. Practical efficiencies are reported in Sect. 5.

Transposing Four 8×8 Byte Matrices Simultaneously. Our next task is to design a transposition algorithm of four packed 8×8 byte matrices simultaneously. More precisely, it performs the following operation on a packed 32×8 byte matrix:

$$\begin{pmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \end{pmatrix} \longrightarrow \begin{pmatrix} M_0^\top \\ M_1^\top \\ M_2^\top \\ M_3^\top \end{pmatrix},$$

where the M_i are 8×8 blocks. This operation has the following prototype in the sequel:

```
void avx_packed_matrix_byte_8x8_transpose
  (avx_uint64_t* dest, const avx_uint64_t* src);
```

This function works as follows. First the input `src` is loaded into eight AVX registers $r_0, \ldots, r_7$. Each r_i is seen as a vector of four `uint64_t`: for $j \in \{0, \ldots, 3\}$, $r_0[j], \ldots, r_7[j]$ thus represent the 8×8 byte matrix M_j. Then we transpose these four matrices simultaneously in-register by means of AVX shift and blend operations over 32, 16 and 8 bits entries in the spirit of the aforementioned divide and conquer strategy.

Transposing 256×64 Bit Matrices. Having the above subroutines at our disposal, we can now present our algorithm to transpose a packed 256×64 bit matrix.

The input bit matrix of type `avx_int64_t` is written $(M_{i,j})_{0\leqslant i<256,\,0\leqslant j<64}$. The transposed output matrix is written $(N_{i,j})_{0\leqslant i<64,\,0\leqslant j<256}$ and has type `uint64_t*`. We first compute the auxiliary byte matrix T as follows:

```
static avx_uint64_t T[64];
for (int i= 0; i < 8; i++) {
  avx_packed_matrix_byte_8x8_transpose (T + 8*i, M + 8*i);
  for (int k= 0; k < 8; k++)
    T[8*i+k]= avx_packed_matrix_bit_8x8_transpose(T[8*i+k]);
```

If we write $M_{i,k:l}$ for the byte representing the packed bit vector $(M_{i,k},\dots,M_{i,l})$, then T contains the following 32×64 byte matrix:

$$\left(\begin{array}{ccc|ccc|c|ccc}
M_{0,0:7} & \dots & M_{56,0:7} & M_{0,8:15} & \dots & M_{56,8:15} & \dots & M_{0,56:63} & \dots & M_{56,56:63} \\
\vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\
M_{7,0:7} & \dots & M_{63,0:7} & M_{7,8:15} & \dots & M_{63,8:15} & \dots & M_{7,56:63} & \dots & M_{63,56:63} \\
\hline
M_{64,0:7} & \dots & M_{120,0:7} & M_{64,8:15} & \dots & M_{120,8:15} & \dots & M_{64,56:63} & \dots & M_{120,56:63} \\
\vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\
M_{71,0:7} & \dots & M_{127,0:7} & M_{71,8:15} & \dots & M_{127,8:15} & \dots & M_{71,56:63} & \dots & M_{127,56:63} \\
\hline
M_{128,0:7} & \dots & M_{184,0:7} & M_{128,8:15} & \dots & M_{184,8:15} & \dots & M_{128,56:63} & \dots & M_{184,56:63} \\
\vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\
M_{135,0:7} & \dots & M_{191,0:7} & M_{135,8:15} & \dots & M_{191,8:15} & \dots & M_{135,56:63} & \dots & M_{191,56:63} \\
\hline
M_{192,0:7} & \dots & M_{248,0:7} & M_{192,8:15} & \dots & M_{248,8:15} & \dots & M_{192,56:63} & \dots & M_{248,56:63} \\
\vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\
M_{199,0:7} & \dots & M_{255,0:7} & M_{199,8:15} & \dots & M_{255,8:15} & \dots & M_{199,56:63} & \dots & M_{255,56:63}
\end{array}\right).$$

First, for all $0\leqslant i\leqslant 7$, we load column $8i$ into the AVX register r_i. We interpret these registers as forming a 32×8 byte matrix that we transpose in-registers. This transposition is again performed in the spirit of the aforementioned divide and conquer strategy and makes use of various specific AVX2 instructions. We obtain

$$\left(\begin{array}{ccc|ccc|c}
M_{0,0:7} & M_{1,0:7} & \dots & M_{7,0:7} & M_{64,0:7} & M_{65,0:7} & \dots & M_{71,0:7} & \dots \\
M_{0,8:15} & M_{1,8:15} & \dots & M_{7,8:15} & M_{64,8:15} & M_{65,8:15} & \dots & M_{71,8:15} & \dots \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \\
M_{0,56:63} & M_{1,56:63} & \dots & M_{7,56:63} & M_{64,56:63} & M_{65,56:63} & \dots & M_{71,56:63} & \dots
\end{array}\right).$$

More precisely, for $i=0,\dots,7$, the group of four consecutive columns from $4i$ until $4i+3$ is in the register r_i. We save the registers $r_0,\dots,r_7$ at the addresses $N, N+4, N+64, N+68, N+128, N+132, N+192$ and $N+196$.

For each $k=1,\dots,7$, we build a similar 32×8 byte matrix from the columns $k, 8+k,\dots,56+k$ of T, and transpose this matrix using the same algorithm.

This time the result is saved at the addresses $N', N'+4, N'+64, N'+68, N'+128, N'+132, N'+192$ and $N'+196$, where $N' = N+8k$. This yields an efficient routine for transposing M into N, whose prototype is given by

```
void packed_matrix_bit_256x64_transpose
  (uint64_t* N, (const avx_uint64_t*) M);
```

4.3 Frobenius Encoding

If the input polynomial A has degree less than $\ell \leqslant 60m$ and is in packed representation, then it can also be seen as a $m \times 60$ matrix in packed representation (except a padding with zeros could be necessary to adjust the size).

In this setting, the polynomials P_i of Algorithm 1 are simply read as the rows of the matrix. Therefore, to compute the Frobenius encoding $F_\omega(A)$, we only need to transpose this matrix, then add 4 rows of zeros for alignment (because we store one element of $\mathbb{F}_{2^{60}}$ per quad word) and multiply by twiddle factors. This leads to the following implementation:

Function 7.
Input: $A(x) = \sum_{0 \leqslant i < \ell} a_i x^i \in \mathbb{F}_2[x]$.
Output: $F_\omega(A)$ stored from pointer d to m allocated quad words.
Assumptions: $n = 61m$ divides $2^{60} - 1$ and $\ell \leqslant 60m$.

```
void encode (uint64_t* d, const uint64_t& m,
             const uint64_t* A, const uint64_t& ℓ) {
1. uint64_t c = 1, i = 0, e = 0;
2. avx_uint64_t v[64]; uint64_t w[256];
3. while (i < m) {
4.   e = min (m - i, 256);
5.   for (int j = 0; j < 64; j++)
       load (v[j], A, ℓ, i + m * j, e);
6.   packed_matrix_bit_256x64_transpose (w, v);
7.   for (int j = 0; j < e; j++) {
       d[i + j] = f2_60_mul (w[j], c);
       c = f2_60_mul (c, ω); }
8.   i += e; }
```

Remark 3. To optimize read accesses, it is better to run loop 5 for $j < \lceil l/m \rceil$ and to initialize the remaining `v[j]` to zero. Indeed, for a product of degree d, we typically multiply two polynomials of degree $\simeq d/2$, which means $\ell < 30m$ when computing the direct transform.

The Frobenius decoding consists in inverting the encoding. The implementation issues are the same, so we refer to our source code for further details.

5 Timings

The platform considered in this paper is equipped with an Intel(R) Core(TM) i7-6700 CPU at 3.40 GHz and 32 GB of 2133 MHz DDR4 memory. This CPU features AVX2, BMI2 and CLMUL technologies (family number 6 and model number 94). The platform runs the Stretch GNU Debian operating system with a 64 bit Linux kernel version 4.3. We compile with GCC [6] version 5.4.

We use version 1.2 of the gf2x library (https://gforge.inria.fr/projects/gf2x/, released in July 2017)—it makes use of the CLMUL features of the platform. We tuned it to our platform during the installation process up to 32000000 input quad words. We also compare to the implementation of the additive Fourier transform by Chen et al. [2], using the GIT version of 2017, September, 1.

Frobenius Encoding. Concerning the cost of the Frobenius encoding and decoding, Function 6 takes about 20 CPU cycles when compiled with the sole `-O3` option. With the additional options `-mtune=native -mavx2 -mbmi2`, the BMI2 version of Remark 2 takes about 16 CPU cycles. The vectorized version of Function 6 transposes four packed 8×8 bit matrices simultaneously in about 20 cycles, which makes an average of 5 cycles per matrix.

It is interesting to examine the performance of the sole transpositions made during the Frobenius encoding and decoding (that is discarding products by twiddle factors in $\mathbb{F}_{2^{60}}$). From sizes of a few kilobytes this average cost per quad word is about 8 cycles with the AVX2 technology, and it is about 23 cycles without. Unfortunately the vectorization speed-up is not as close to 4 as we would have liked.

Since the encoding and decoding costs are linear, their relative contribution to the total computation time of polynomial products decreases for large sizes. For two input polynomials in $\mathbb{F}_2[x]$ of 2^{16} quad words, the contribution is about 15%; for 2^{22} quad words, it is about 10%.

Polynomial Product. In Fig. 1 we report timings in milliseconds for multiplying two polynomials in $\mathbb{F}_2[x]_{<\ell}$, hence each of input size $\lceil \ell/64 \rceil$ quad words—indicated in abscissa and obtained from `justinline/bench/polynomial_f2_bench.mmx`. Notice that our implementation in [7] was faster than version 1.1 of gf2x, but is now of similar speed as version 1.2. The additive FFT strategy of [2] achieves a noticeable speed-up in favorable cases, but because of its staircase-effect its runtime is roughly similar to the one of gf2x in average. With respect to our old implementation, the new one finally achieves a speed-up that is not far from the factor 2 predicted by the asymptotic complexity analysis. Let us mention that our new implementation becomes faster than gf2x when $\lceil \ell/64 \rceil$ is larger than 2048.

Polynomial Matrix Product. As in [7], one major advantage of DFTs over the Babylonian field $\mathbb{F}_{2^{60}}$ is the compactness of the evaluated FFT-representation of polynomials. This makes linear algebra over $\mathbb{F}_2[x]$ particularly efficient: instead

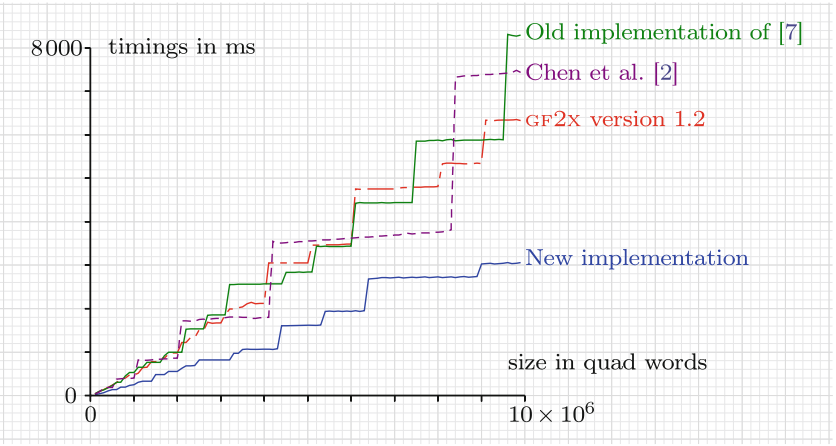


Fig. 1. Products in $\mathbb{F}_2[x]_{<\ell}$, input size $\lceil \ell/64 \rceil$ quad words, timings in milliseconds.

of multiplying $r \times r$ matrices over $\mathbb{F}_2[x]_{<\ell}$ naively by means of r^3 polynomial products of degree $< \ell$, we use the standard evaluation-interpolation approach. In our context, this comes down to: (a) computing the $2r^2$ Frobenius encodings, (b) the $2r^2$ direct DFTs of all entries of the two matrices to be multiplied, (c) performing the $\approx 2\ell/60$ products of $r \times r$ matrices over $\mathbb{F}_{2^{60}}$, (d) computing the r^2 inverse DFTs and Frobenius decodings of the so-computed matrix products.

Timings for matrices over $\mathbb{F}_2[x]$ are obtained from `justinline/bench/matrix_polynomial_f2_bench.mmx` and are reported in Table 1. The row "this paper" confirms the practical gain of this fast approach within our implementation. For comparison, the row "GF2X" shows the cost of computing the product naively, by doing r^3 polynomial multiplications using GF2X. More efficient evaluation-interpolation based approaches [10, Sect. 2] for matrix multiplication can in principle be combined with Schönhage's triadic polynomial multiplication [16] as implemented in GF2X. However, this would require an additional implementation effort and also lead to an extra constant overhead with respect to our approach.

Table 1. Products of $r \times r$ matrices over $\mathbb{F}_2[x]$, for degree $64 \cdot 2^{16}$, in milliseconds

r	1	2	4	8	16	32
This paper	12	51	212	896	3969	18953
GF2X	22	182	1457	11856	92858	745586

6 Conclusion

The present paper describes a major new approach for the efficient computation of large carryless products. It confirms the excellent arithmetic properties of the Babylonian field $\mathbb{F}_{2^{60}}$ for practical purposes, when compared to the fastest previously available strategies.

Improvements are still possible for our implementation of DFTs over $\mathbb{F}_{2^{60}}$. First, taking advantage of the more recent AVX-512 technologies is an important challenge. This is difficult due to the current lack of 256 or 512 bit SIMD counterparts for the `vpclmulqdq` assembly instruction (carryless multiplication of two quad words). However, larger vector instruction would be beneficial for matrix transposition, and even more taking into account that there are twice as many 512 bit registers as 256 bit registers; so we can expect a significant speed-up for the Frobenius encoding/decoding stages. The second expected improvement concerns the use of truncated Fourier transforms [9,14] in order to smoothen the graph from Fig. 1. Finally we expect that our new ideas around the Frobenius transform might be applicable to other small finite fields.

References

1. Brent, R.P., Gaudry, P., Thomé, E., Zimmermann, P.: Faster multiplication in GF(2)[x]. In: van der Poorten, A.J., Stein, A. (eds.) ANTS 2008. LNCS, vol. 5011, pp. 153–166. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79456-1_10
2. Chen, M.-S., Cheng, C.-M., Kuo, P.-C., Li, W.-D., Yang, B.-Y.: Faster multiplication for long binary polynomials (2017). https://arxiv.org/abs/1708.09746
3. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Math. Comput. **19**, 297–301 (1965)
4. Gao, S., Mateer, T.: Additive fast Fourier transforms over finite fields. IEEE Trans. Inform. Theory **56**(12), 6265–6272 (2010)
5. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra, 3rd edn. Cambridge University Press, New York (2013)
6. GCC, the GNU Compiler Collection (1987). http://gcc.gnu.org
7. Harvey, D., van der Hoeven, J., Lecerf, G.: Fast polynomial multiplication over $\mathbb{F}_{2^{60}}$. In: Rosenkranz, M. (ed.) Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016, pp. 255–262. ACM (2016)
8. Harvey, D., van der Hoeven, J., Lecerf, G.: Faster polynomial multiplication over finite fields. J. ACM **63**(6) (2017). Article 52
9. van der Hoeven, J.: The truncated Fourier transform and applications. In: Schicho, J. (ed.) Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation, ISSAC 2004, pp. 290–296. ACM (2004)
10. van der Hoeven, J.: Newton's method and FFT trading. J. Symbolic Comput. **45**(8), 857–878 (2010)
11. van der Hoeven, J., Larrieu, R.: The Frobenius FFT. In: Burr, M. (ed.) Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2017, pp. 437–444. ACM (2017)

12. van der Hoeven, J., Lecerf, G.: Interfacing Mathemagix with C++. In: Monagan, M., Cooperman, G., Giesbrecht, M. (eds.) Proceedings of the 2013 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2013, pp. 363–370. ACM (2013)
13. van der Hoeven, J., Lecerf, G.: Mathemagix User Guide (2013). https://hal.archives-ouvertes.fr/hal-00785549
14. Larrieu, R.: The truncated Fourier transform for mixed radices. In: Burr, M. (ed.) Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2017, pp. 261–268. ACM (2017)
15. Lin, S.-J., Chung, W.-H., Yunghsiang Han, S.: Novel polynomial basis and its application to Reed-Solomon erasure codes. In: 2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS), pp. 316–325. IEEE (2014)
16. Schönhage, A.: Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. Acta Infor. **7**, 395–398 (1977)
17. Schönhage, A., Strassen, V.: Schnelle Multiplikation großer Zahlen. Computing **7**, 281–292 (1971)
18. Warren, H.S.: Hacker's Delight, 2nd edn. Addison-Wesley, Boston (2012)

Improving Enclosure of Interval Scalar Projection Operation

Tomasz Dobrowolski(✉)

Gdańsk University of Technology, Gdańsk, Poland
tomasz.dobrowolski@live.com

Abstract. We introduce interval scalar projection operation with tight interval enclosure. Our approach relies on the solution to non-convex optimization problem. We present an improved algorithm for computing interval scalar projection for 2-dimensional box intervals and compare to a simple algorithm based on natural interval extension method. Applications include automated verification of properties of geometric algorithms and computing Voronoi diagrams over inexact input data.

Keywords: Interval arithmetic · Scalar projection
Non-convex optimization · Geometric algorithms verification
Computational geometry · Voronoi diagrams

1 Introduction

The *scalar projection* $s(\boldsymbol{x_1}, \boldsymbol{x_2})$ is an inner product of $\boldsymbol{x_1}$ and $\widehat{\boldsymbol{x_2}}$, where $\boldsymbol{x_1} \in \mathbb{R}^n$ and $\widehat{\boldsymbol{x_2}}$ is the unit vector in the direction of $\boldsymbol{x_2} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}$.

One way to define *interval scalar projection* for n-dimensional box intervals is to use its *natural interval extension* [8], i.e. using elementary interval arithmetic operations such as interval multiplication, interval addition, interval square root and interval division. Such straightforward computation contributes to an unnecessary overestimation, since it suffers from dependency effects [8,9]. Moreover, at every step the intermediate result has to be promoted to a box interval, which is especially problematic for interval normalization.

Over many years of research in interval analysis, numerous methods were developed to improve enclosures of interval extensions, including interval splitting [8], affine arithmetic [17], Taylor methods [10] and more [3,11].

In automated theorem proving and verification systems [12,19], e.g. Coq, PVS, the excessive overestimation can significantly slow down the verification process or prevent it from completion.

In this article we propose a method for computing tight lower and upper bounds of interval scalar projection $[s](X_1, X_2)$. Our approach is to express $[s]$ as an optimization problem. We present an improved algorithm for computing interval scalar projection for 2-dimensional box intervals and compare it to straightforward (natural interval extension) implementation.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 137–146, 2017.
https://doi.org/10.1007/978-3-319-72453-9_10

2 Interval Extensions

Let $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{R}^n$, $\boldsymbol{b} = (b_1, b_2, \ldots, b_n) \in \mathbb{R}^n$. Interval $[\boldsymbol{a}, \boldsymbol{b}]$ is an n-dimensional *box interval* in *interval space* $\mathbb{IR}^n$, if

$$[\boldsymbol{a}, \boldsymbol{b}] = \prod_{1 \leq i \leq n} [a_i b_i], \quad a_i \leq b_i. \tag{1}$$

For $X = [\boldsymbol{a}, \boldsymbol{b}]$, we denote $\underline{X} = \boldsymbol{a}$ as a lower bound and $\overline{X} = \boldsymbol{b}$ as an upper bound of X.

A function $f(\boldsymbol{x_1}, \ldots, \boldsymbol{x_k})$, where $\boldsymbol{x_1}, \ldots, \boldsymbol{x_k} \in \mathbb{R}^n$, can be extended to a function

$$f(X_1, \ldots, X_k) = \{f(\boldsymbol{x_1}, \ldots, \boldsymbol{x_k}) \mid \forall_{1 \leq i \leq k}\ \boldsymbol{x_i} \in X_i\}, \tag{2}$$

where $X_1, \ldots, X_k \subseteq \mathbb{R}^n$.

Let $[f]$ denote an *interval extension* of f, if for every $X_1, \ldots, X_k \in \mathbb{IR}^n$

$$[f](X_1, \ldots, X_k) \supseteq f(X_1, \ldots, X_k). \tag{3}$$

We want enclosure of the interval extension to be as tight as possible.

3 Interval Scalar Projection in $\mathbb{R}^n$

We say $\widehat{\boldsymbol{x}}$ is a *unit vector* in the direction of $\boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}$, if $\widehat{\boldsymbol{x}} = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$, where $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$ (Euclidean ℓ_2 norm).

$\widehat{X}$ is a set of unit vectors in the direction of elements in $X \subseteq \mathbb{R}^n \setminus \{\boldsymbol{0}\}$, if

$$\widehat{X} = \left\{ \widehat{\boldsymbol{x}} \;\middle|\; \boldsymbol{x} \in X \right\}. \tag{4}$$

Note that $\widehat{X}$ is a subset of a unit sphere, denoted $\widehat{X} \subseteq \mathbb{S}^n$.

We say $s : \mathbb{R}^n \times (\mathbb{R}^n \setminus \{\boldsymbol{0}\}) \to \mathbb{R}$ is a *scalar projection*, if

$$s(\boldsymbol{x_1}, \boldsymbol{x_2}) = \langle \boldsymbol{x_1}, \frac{\boldsymbol{x_2}}{\|\boldsymbol{x_2}\|} \rangle. \tag{5}$$

A scalar projection in Euclidean space can be expressed in terms of cosine of the angle, that is,

$$s(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1\| \cos \angle(\boldsymbol{x}_1, \boldsymbol{x}_2), \tag{6}$$

where $\angle(\boldsymbol{x}_1, \boldsymbol{x}_2) \in [0, \pi]$ is the smallest angle between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

For any $\boldsymbol{x}_1 \in \mathbb{R}^n$ and $\boldsymbol{x}_2 \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}$,

$$-\|\boldsymbol{x}_1\| \leq s(\boldsymbol{x}_1, \boldsymbol{x}_2) \leq \|\boldsymbol{x}_1\|. \tag{7}$$

Let $X_1, X_2 \in \mathbb{IR}^n$, $\boldsymbol{0} \notin X_2$.

An *interval scalar projection* is an interval extension $[s](X_1, X_2)$ of $s(X_1, X_2)$.

The problem of finding minimum and maximum of $s(X_1, X_2)$ is a non-convex optimization problem and has the form

$$\begin{array}{ll} \text{minimize(maximize)} & s(\boldsymbol{x_1}, \boldsymbol{x_2}) \\ \text{subject to} & \boldsymbol{x_1} \in X_1, \boldsymbol{x_2} \in X_2. \end{array} \tag{8}$$

Proposition 1. *For $X_1, X_2 \in \mathbb{IR}^n$, $\mathbf{0} \notin X_2$, the minimum of $s(X_1, X_2)$ is equal to the negation of the maximum of $s(-X_1, X_2)$.*

Proof. The problem of minimization of $s(\boldsymbol{x_1}, \boldsymbol{x_2})$ can be solved as maximization of $-s(\boldsymbol{x_1}, \boldsymbol{x_2})$, and by the definition of the inner product for any $\boldsymbol{x_1} \in \mathbb{R}^n$ and $\widehat{\boldsymbol{x_2}} \in \mathbb{S}^n$, $-\langle \boldsymbol{x_1}, \widehat{\boldsymbol{x_2}} \rangle = \langle -\boldsymbol{x_1}, \widehat{\boldsymbol{x_2}} \rangle$.

3.1 Lower and Upper Bounds of Interval Scalar Projection in $\mathbb{R}^n$

Let $g(X)$ be the set of all vertices of a box interval $X \in \mathbb{IR}^n$.

For $X = [(a_1, a_2, \ldots, a_n), (b_1, b_2, \ldots, b_n)]$,

$$g(X) = \{(x_1, x_2, \ldots, x_n) \mid x_i = a_i \vee x_i = b_i\}. \tag{9}$$

Note that $|g(X)| = 2^n$ and X is a convex polytope of $g(X)$, that is, $X = \operatorname{conv} g(X)$.

Lemma 1. *For every convex combination $\lambda_1 \boldsymbol{a_1} + \ldots + \lambda_k \boldsymbol{a_k}$ of $\{\boldsymbol{a_1}, \ldots, \boldsymbol{a_k}\}$, where $a_i \in \mathbb{R}^n$, and for a fixed $\boldsymbol{b} \in \mathbb{R}^n$,*

$$\min(\langle \boldsymbol{a_1}, \boldsymbol{b} \rangle, \ldots, \langle \boldsymbol{a_k}, \boldsymbol{b} \rangle) \leq \langle \lambda_1 \boldsymbol{a_1} + \ldots + \lambda_k \boldsymbol{a_k},\ \boldsymbol{b} \rangle \leq \max(\langle \boldsymbol{a_1}, \boldsymbol{b} \rangle, \ldots, \langle \boldsymbol{a_k}, \boldsymbol{b} \rangle).$$

Proof. Let $G = \{\langle \boldsymbol{a_1}, \boldsymbol{b} \rangle, \ldots, \langle \boldsymbol{a_k}, \boldsymbol{b} \rangle\}$ and $f(\lambda_1, \ldots, \lambda_k) = \langle \lambda_1 \boldsymbol{a_1} + \ldots + \lambda_k \boldsymbol{a_k},\ \boldsymbol{b} \rangle$.

After expanding $f(\lambda_1, \ldots, \lambda_k)$ to

$$\lambda_1 \langle \boldsymbol{a_1}, \boldsymbol{b} \rangle + \ldots + \lambda_k \langle \boldsymbol{a_k}, \boldsymbol{b} \rangle$$

we can see that $f(\lambda_1, \ldots, \lambda_k)$ is a convex combination of G. And clearly $\operatorname{conv} G = \operatorname{conv} \{\min G,\ \max G\}$ for every finite set of real numbers G (the detailed derivation is omitted).

Lemma 2. *Let s be a scalar projection in $\mathbb{R}^n$. For every $A, B \in \mathbb{IR}^n$, $\mathbf{0} \notin B$, $s(A, B) = s(g(A), B)$.*

Proof. Scalar projection function is continuous in $\mathbb{R}^n \times (\mathbb{R}^n \setminus \{\mathbf{0}\})$, therefore for $\mathbf{0} \notin B$ its interval extension $s(A, B) = s(g(A), B) \iff \min s(A, B) = \min s(g(A), B) \wedge \max s(A, B) = \max s(g(A), B)$.

It is equivalent to demonstrating that for a fixed $\widehat{\boldsymbol{b}} \in \mathbb{S}^n$ and for every $\boldsymbol{a} \in A$,

$$\min s(g(A), \{\widehat{\boldsymbol{b}}\}) \leq s(\boldsymbol{a}, \widehat{\boldsymbol{b}}) \leq \max s(g(A), \{\widehat{\boldsymbol{b}}\}),$$

which directly follows from Lemma 1, since $\boldsymbol{a}$ is a convex combination of $g(A)$.

Proposition 2. *To find maximum (or minimum) of $s(A, B)$ it is enough to find maximum (minimum) of s for all the corners of A, i.e.*

$$\max s(A, B) = \max\{\max s(\{\boldsymbol{x}\}, B) \mid \boldsymbol{x} \in g(A)\}.$$

Proof. Directly follows from Lemma 2.

Lemma 3. *For a given* $\boldsymbol{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$, $X \in \mathbb{IR}^n$, $\mathbf{0} \notin X$, *the maximum of* $s(\{\boldsymbol{x}\}, X)$ *is equal to* $s(\boldsymbol{x}, \boldsymbol{y})$ *for any vector* $\boldsymbol{y} \in X$ *that is at the minimum angle to* $\boldsymbol{x}$, *i.e.*

$$\max s(\{\boldsymbol{x}\}, X) = s\left(\boldsymbol{x},\ \operatorname*{arg\,min}_{\boldsymbol{y} \in X} \angle(\boldsymbol{x}, \boldsymbol{y})\right).$$

Proof. A scalar projection in Euclidean space can be expressed in terms of cosine of the angle (see Eq. 6).

The proof follows directly from the fact cos is monotonically decreasing in $[0, \pi]$.

Let pos G denote a set of conic combinations of a finite set $G \subset \mathbb{R}^n$, with a boundary ∂pos G. If $\mathbf{0} \notin$ conv G, then pos G is non-trivial and convex polyhedral (see [16,18]).

Proposition 3. *For a given box interval* $X \in \mathbb{IR}^n$, $\mathbf{0} \notin X$, *and* $\boldsymbol{x} \in \mathbb{R}^n$,

$$\max s(\{\boldsymbol{x}\}, X) = \begin{cases} \|\boldsymbol{x}\| & \boldsymbol{x} \in \mathit{pos}\ g(X) \\ s\left(\boldsymbol{x}, \operatorname*{arg\,min}_{\boldsymbol{y} \in \partial \mathit{pos}\ g(X), \boldsymbol{y} \neq \mathbf{0}} \angle(\boldsymbol{x}, \boldsymbol{y})\right) & \boldsymbol{x} \notin \mathit{pos}\ g(X). \end{cases}$$

Proof. Note that $\boldsymbol{x} = \mathbf{0}$ implies $\boldsymbol{x} \in$ pos $g(X)$, since $\mathbf{0} \in$ pos $g(X)$, and for any $\boldsymbol{y} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$, $\max s(\mathbf{0}, \boldsymbol{y}) = 0$.

In other cases, we follow Lemma 3.

If $\boldsymbol{x} \neq \mathbf{0}$ and $\boldsymbol{x} \in$ pos $g(X)$, the minimal angle is equal to 0, therefore $\max s(\{\boldsymbol{x}\}, X) = \|\boldsymbol{x}\| \cos(0) = \|\boldsymbol{x}\|$.

Note that pos $g(X)$ is non-trivial and convex polyhedral, since $g(X)$ is finite and $\mathbf{0} \notin X$, $X =$ conv $g(X)$. Therefore, if $\boldsymbol{x} \notin$ pos $g(X)$, it is enough to search for $\boldsymbol{y}$ at the boundary of pos $g(X)$ (excluding the origin).

The problem of finding $\max s(\{\boldsymbol{x}\}, X)$ can be solved using a solution to a more general problem of finding critical angles between two convex cones [14], except in our case one cone is a single ray (pos $\{\boldsymbol{x}\}$).

In R^3 ($\boldsymbol{x} \in \mathbb{R}^3$ and $X \in \mathbb{IR}^3$, $\mathbf{0} \notin X$) we can use the fact that an argument $\boldsymbol{y} \in \partial$pos $g(X)$ ($\boldsymbol{y} \neq \mathbf{0}$) between pos $\{\boldsymbol{x}\}$ and pos $g(X)$ is either an orthogonal projection of $\boldsymbol{x}$ onto ∂pos $g(X)$ (the closest point of $\boldsymbol{x}$ to ∂pos $g(X)$) or it lies on one of the extreme rays of pos $g(X)$.

In R^2, if $\boldsymbol{y} \notin$ pos $g(X)$, it is enough to calculate minimum of scalar projections of $\boldsymbol{x}$ to extreme corners of X that are intersecting ∂pos $g(X)$.

The general problem of finding orthogonal projection of a vector onto a convex cone can be solved using Dykstra's algorithm [2,6].

4 Computation of Interval Scalar Projection in $\mathbb{R}^2$

In our implementations we are assuming that a library of elementary interval operations is provided with following operations on 1-dimensional (real number)

intervals: addition (INTADD), subtraction (INTSUB), multiplication (INTMUL), division (INTDIV), square (INTSQ) and square root (INTSQRT). We are assuming implementation of those operations already have tight enclosures for a given floating-point precision. Implementation of such operations and verification of their correctness is discussed at length in [5].

4.1 Natural Interval Extension Method

A straightforward implementation of interval scalar projection in $\mathbb{R}^2$ can be simply expressed using elementary interval arithmetic operations instead of usual point-arithmetic operations (see Algorithm 1). This is often called *natural interval extension method* [8].

Algorithm 1. Simple algorithm computing interval scalar projection in $\mathbb{R}^2$

```
function intNorm([a, b])
    P ← intSq([a_x, b_x])          ▷ intSq denotes interval square operation
    Q ← intSq([a_y, b_y])
    R ← intAdd(P, Q)               ▷ intAdd denotes interval addition
    return intSqrt(R)              ▷ intSqrt denotes interval square root
end function

function intProd([a, b],[c, d])
    P ← intMul([a_x, b_x], [c_x, d_x])   ▷ intMul denotes interval multiplication
    Q ← intMul([a_y, b_y], [c_y, d_y])
    return intAdd(P, Q)
end function

function intScalarProj(A,B)                ▷ A ∈ IR^2, B ∈ IR^2, 0 ∉ B
    N ← intNorm(B)
    B̂ ← intDiv(B, N)
    return intProd(A, B̂)
end function
```

4.2 Improved Algorithm

An improved algorithm is based on the solution to the optimization problem stated in Eq. 8.

Let $A \in \mathbb{IR}^2$, $B \in \mathbb{IR}^2$, $\mathbf{0} \notin B$ be a pair of input box intervals.

Let $g(A)$ be the set of corners of A,

$$g(A) = \{(\underline{A}_x, \underline{A}_y), (\underline{A}_x, \overline{A}_y), (\overline{A}_x, \underline{A}_y), (\overline{A}_x, \overline{A}_y)\} \tag{10}$$

Let $e(A) = (\boldsymbol{e_1}, \boldsymbol{e_2})$ be the ordered set (in clock-wise order) of extreme corners of A touching the boundary of pos $g(A)$, such that pos $e(A)$ = pos $g(A)$, $e(A) \cap$

Algorithm 2. Improved algorithm computing interval scalar projection in $\mathbb{R}^2$

```
function intSignedArea([a, b],[c, d])
    P ← intMul([a_x, b_x], [c_y, d_y])
    Q ← intMul([a_y, b_y], [c_x, d_x])
    return intSub(P, Q)
end function

function insideCone(x,e_1,e_2)
    P ← intSignedArea([e_1, e_1], [x, x])
    Q ← intSignedArea([x, x], [e_2, e_2])
    return P̄ ≥ 0 ∧ Q̄ ≥ 0
end function

function maximize(x,B)                          ▷ x ∈ ℝ², B ∈ 𝕀ℝ², 0 ∉ B
    (e_1,e_2) ← e(B)        ▷ e(B) is set of extreme corners of B in clock-wise order
    if insideCone(x, e_1, e_2) then
        R ← intNorm([x, x])
        return R̄
    else
        r_max ← −∞
        for all y ∈ {e_1,e_2} do
            R ← intScalarProj([x, x], [y, y])
            r_max ← max(r_max, R̄)
        end for
        return r_max
    end if
end function

function intScalarProjImproved(A,B)             ▷ A ∈ 𝕀ℝ², B ∈ 𝕀ℝ², 0 ∉ B
    r_min ← +∞
    r_max ← −∞
    for all x ∈ g(A) do                         ▷ g(A) is set of corners of A
        r_min ← min(r_min, −maximize(−x, B))
        r_max ← max(r_max, maximize(x, B))
    end for
    return [r_min, r_max]
end function
```

$\partial\text{pos}\ g(A) = e(A)$, $e(A) \subset g(A)$ and signed area of a triangle $(\mathbf{0}, \boldsymbol{e_1}, \boldsymbol{e_2})$ is non-negative. If $\mathbf{0} \notin A$ and A is non-degenerate (has width greater then zero), there can be at most 2 extreme corners. For degenerate A (when A is a point) all corners are equal, therefore $\boldsymbol{e_1} = \boldsymbol{e_2}$. To find extreme corners it is enough to consider 8 possible cases of box A crossing the Cartesian coordinate axes on the plane.

We consider computation of $g(A)$ and $e(A)$ as a trivial implementation detail that is omitted in the pseudo-code.

As we have demonstrated in Proposition 2, we can solve global optimization problem for $s(A, B)$ by computing maximum (minimum) of $s(\{\boldsymbol{x}\}, B)$ for every corner $\boldsymbol{x} \in g(A)$. In Algorithm 2, the main function IntScalarProjImproved is iterating over $g(A)$, and for every $\boldsymbol{x} \in g(A)$ computes a minimum of $s(\{\boldsymbol{x}\}, B)$ (expressed as $-\max s(\{-\boldsymbol{x}\}, B)$, see Proposition 1) and maximum of $s(\{\boldsymbol{x}\}, B)$.

The maximization of $s(\{\boldsymbol{x}\}, B)$ is implemented in the function maximize. As in Proposition 3 we consider two cases, when $\boldsymbol{x}$ is inside or outside of pos $\{\boldsymbol{e_1}, \boldsymbol{e_2}\} = \text{pos } g(B)$.

The function insideCone is estimating signed area of two triangles $(\mathbf{0}, \boldsymbol{e_1}, \boldsymbol{x})$ and $(\mathbf{0}, \boldsymbol{x}, \boldsymbol{e_2})$. In point arithmetic, to check if $\boldsymbol{x}$ is inside the cone, it would be enough to check if areas of those two triangles are non-negative. To compensate for floating-point precision inaccuracies, we check if the upper bounds of area intervals are non-negative (intSignedArea returns an interval). Consequently, in some cases insideCone may return true even if $\boldsymbol{x}$ is outside (but close to the cone boundary). This may contribute to overestimation of the final result, but will never lead to underestimation, because $\|\boldsymbol{x}\|$ (estimated by intNorm($[\boldsymbol{x}, \boldsymbol{x}]$)) is always greater or equal than any scalar projection of $\boldsymbol{x}$ (see Eq. 7).

If $\boldsymbol{x}$ is (strongly) outside of pos $\{\boldsymbol{e_1}, \boldsymbol{e_2}\}$, we calculate maximum of scalar projection of $\boldsymbol{x}$ onto all extreme corners in $\{\boldsymbol{e_1}, \boldsymbol{e_2}\}$. The upper bound of scalar projection is calculated simply by calling intScalarProj function from Algorithm 1. However, this time it is computed for degenerate intervals (points), therefore, it will not suffer from excessive overestimation.

In fact, in our implementation we commonly use interval operations for degenerate intervals (points) only to compensate for inaccuracies of floating-point arithmetic.

4.3 Interval Enclosure Quality Comparison

In Fig. 1 we compare how two algorithms perform for examplary box intervals. We can clearly see that Algorithm 2 (improved) provides tighter interval enclosure across the whole domain. From our experiments, the overestimation of resulting interval in Algorithm 1 is (significantly) increasing when B is moving closer to the origin, while Algorithm 2 remains numerically stable (unless $[\mathbf{0} - \varepsilon, \mathbf{0} + \varepsilon] \in B$, where ε depends on floating-point precision and accuracy of rational square root approximation).

5 Applications

Scalar projection operation can be used for computing closest distances between points and hyperplanes (lines in $\mathbb{R}^2$).

A *bisector* $\mathcal{B}(\boldsymbol{a}, \boldsymbol{b})$ for $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ is a hyperplane equidistant to $\boldsymbol{a}$ and $\boldsymbol{b}$.

Let's express signed ℓ_2 distance between a point $\boldsymbol{x} \in \mathbb{R}^n$ and a *bisector* $\mathcal{B}(\boldsymbol{a}, \boldsymbol{b})$ using scalar projection function s,

$$\delta_{\mathcal{B}}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) = \left\langle \frac{\boldsymbol{a}+\boldsymbol{b}}{2} - \boldsymbol{x},\ \frac{\boldsymbol{b}-\boldsymbol{a}}{\|\boldsymbol{b}-\boldsymbol{a}\|} \right\rangle = s\left(\frac{\boldsymbol{a}+\boldsymbol{b}}{2} - \boldsymbol{x},\ \boldsymbol{b}-\boldsymbol{a} \right), \tag{11}$$

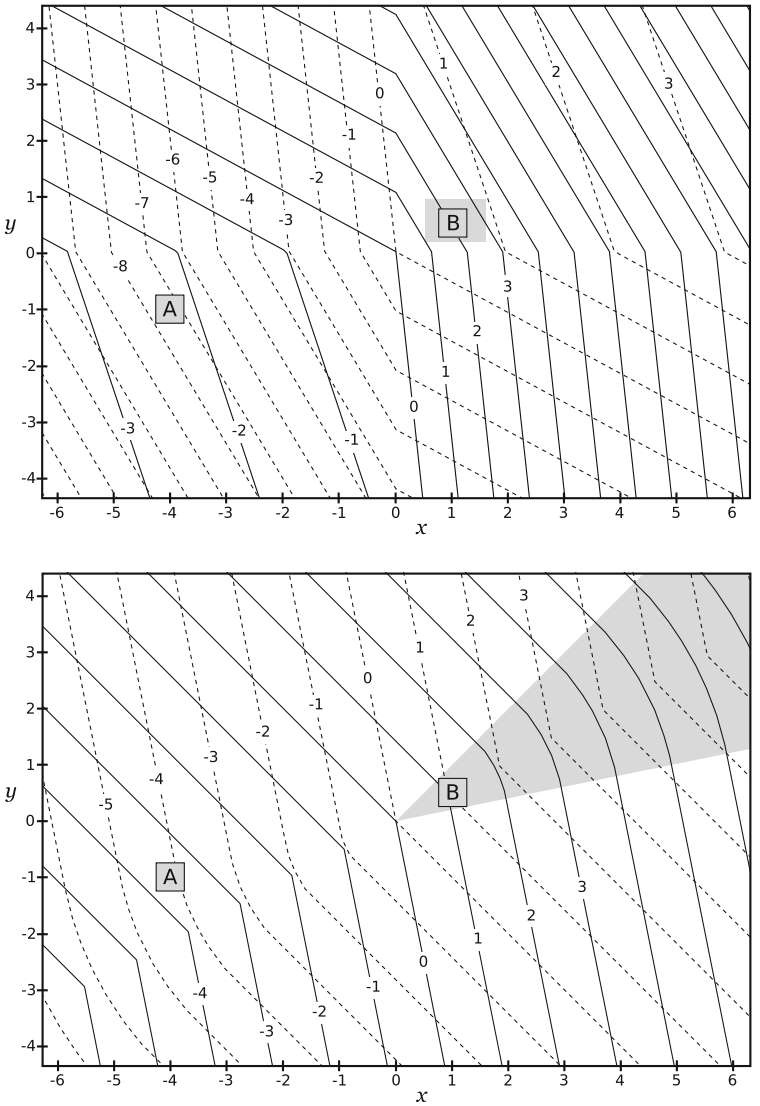


Fig. 1. Contour plot of lower and upper bounds of $[s](\{(x,y)\},B)$ for natural interval extension method (top) and proposed improved method (bottom), where $B = [0.75, 1.25] \times [0.25, 0.75]$. Contour lines of lower bounds are dashed. In natural interval extension method (top) we shade area around B that represents (overestimated) normalized interval $\text{INTDIV}(B, \text{INTNORM}(B))$. While in improved method (bottom) we shade pos $g(B)$, which is the smallest cone enclosing B. Let's compare overestimation for a given interval $A = [-4.25, -3.75] \times [-1.25, -0.75]$. The first algorithm returns $[s]_1(A,B) = [-7.906, -2.057]$, while the second (improved) $[s]_2(A,B) = [-4.431, -3.180]$. Note that $[s]_1(A,B)$ is heavily overestimated and $[s]_2(A,B) \subset [s]_1(A,B)$.

where $\frac{\boldsymbol{a}+\boldsymbol{b}}{2} \in \mathcal{B}(\boldsymbol{a}, \boldsymbol{b})$ and $\frac{\boldsymbol{b}-\boldsymbol{a}}{\|\boldsymbol{b}-\boldsymbol{a}\|}$ is a normal vector perpendicular to $\mathcal{B}(\boldsymbol{a}, \boldsymbol{b})$.

An interval extension $[\delta_{\mathcal{B}}]$ of $\delta_{\mathcal{B}}$ can be simply expressed using interval extension of scalar projection,

$$[\delta_{\mathcal{B}}](X, A, B) = [s]\left(\textsc{intSub}\left(\frac{\textsc{intAdd}(A, B)}{2}, X\right), \textsc{intSub}(B, A)\right). \quad (12)$$

Many problems in computational geometry involve comparing distances between a pair of geometric features, e.g. variants of nearest neighbor search algorithms [1,15,20].

Let's consider problem of finding closest point site in $S \subset \mathbb{R}^n$ to a given query point in $\mathbb{R}^n$, often called Post-Office Location Problem [15]. Additionally, let's assume sites are densely populated over a regular grid $\mathcal{G}^n$, such that every grid cell in $\mathcal{G}^n$ contains at least one site in S (similarly to [20]). Using interval analysis, i.e. interval splitting and branch-and-bound methods [8], we can verify computationally various properties of the algorithm. An example of such property is the number of cells that have to be visited in the worst case in order to find closest site in S for any query point in $\mathbb{R}^n$. We can verify similar properties in extended version of the problem, when sites are represented as line segments. Tight enclosures of interval operations help to reduce the search space.

Other potential applications are algorithms for computing Voronoi diagrams over inexact input data, also called *partial Voronoi Diagrams* [4,7,13].

A study of *partial perpendicular bisectors*, an interesting generalization of interval bisectors, can be found in [7].

6 Conclusion and Future Work

We have presented an improved algorithm in R^2 for computing interval scalar projection with a tight interval enclosure and discussed potential applications.

The next step is to extend our method to R^3 (and R^n for $n > 3$) using theoretical framework and ideas presented in Sect. 3.

Eventually, we plan to develop a libarary that will contain a collection of essential interval operations for automated verification of properties of a broad class of algorithms in computational geometry.

References

1. Aurenhammer, F.: Voronoi diagrams - a survey of a fundamental geometric data structure. ACM Comput. Surv. **23**(3), 345–405 (1991)
2. Bauschke, H.H., Borwein, J.M.: Dykstra's alternating projection algorithm for two sets. J. Approx. Theory **79**(3), 418–443 (1994)
3. Carrizosa, E., Hansen, P., Messine, F.: Improving interval analysis bounds by translations. J. Global Optim. **29**(2), 157–172 (2004)
4. Cheng, R., Xie, X., Yiu, M.L., Chen, J., Sun, L.: UV-diagram: a Voronoi diagram for uncertain data. In: 2010 IEEE 26th International Conference on Data Engineering (ICDE), pp. 796–807. IEEE (2010)

5. Daumas, M., Lester, D., Munoz, C.: Verified real number calculations: a library for interval arithmetic. IEEE Trans. Comput. **58**(2), 226–237 (2009)
6. Dykstra, R.L.: An algorithm for restricted least squares regression. J. Am. Stat. Assoc. **78**(384), 837–842 (1983)
7. Khanban, A.A.: Basic algorithms in computational geometry with imprecise input. Ph.D. thesis, University of London (2005)
8. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. SIAM, Philadelphia (2009)
9. Nedialkov, N.S., Kreinovich, V., Starks, S.A.: Interval arithmetic, affine arithmetic, Taylor series methods: why, what next? Numer. Algorithms **37**(1), 325–336 (2004)
10. Neumaier, A.: Taylor forms–use and limits. Reliable Comput. **9**(1), 43–79 (2003)
11. Neumaier, A.: Improving interval enclosures. Reliable Comput. (2009)
12. Owre, S., Rushby, J.M., Shankar, N.: PVS: a prototype verification system. In: Kapur, D. (ed.) CADE 1992. LNCS, vol. 607, pp. 748–752. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55602-8_217
13. Reem, D.: The geometric stability of Voronoi diagrams with respect to small changes of the sites. In: Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry, SoCG 2011, pp. 254–263. ACM (2011)
14. Seeger, A., Sossa, D.: Critical angles between two convex cones. TOP **24**(1), 44–87 (2016)
15. Shakhnarovich, G., Darrell, T., Indyk, P.: Nearest-Neighbor Methods in Learning and Vision: Theory and Practice. Neural Information Processing. The MIT Press, Cambridge (2006)
16. Stoer, J., Witzgall, C.: Convexity and Optimization in Finite Dimensions I, vol. 163. Springer Science & Business Media, Heidelberg (2012)
17. Stolfi, J., De Figueiredo, L.: An introduction to affine arithmetic. Trends Appl. Comput. Math. **4**(3), 297–312 (2003)
18. Tenenhaus, M.: Canonical analysis of two convex polyhedral cones and applications. Psychometrika **53**(4), 503–524 (1988)
19. Wiedijk, F.: The Seventeen Provers of the World: Foreword by Dana S. Scott, vol. 3600. Springer, Heidelberg (2006)
20. Worley, S.: A cellular texture basis function. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, pp. 291–294. ACM (1996)

Integrating Algebraic and SAT Solvers

Jan Horáček[1(✉)], Jan Burchard[2], Bernd Becker[2], and Martin Kreuzer[1]

[1] Faculty of Informatics and Mathematics, University of Passau, 94030 Passau, Germany
{Jan.Horacek,Martin.Kreuzer}@uni-passau.de
[2] Computer Architecture Group, Albert-Ludwigs-University Freiburg, 79110 Freiburg, Germany
{burchard,becker}@informatik.uni-freiburg.de

Abstract. For solving systems of Boolean polynomials whose zeros are known to be contained in $\mathbb{F}_2^n$, algebraic solvers such as the Boolean Border Basis Algorithm (BBBA) and SAT solvers use very different and possibly complementary methods to create new information. Based on suitable implementations of these solvers and conversion methods from Boolean polynomials to SAT clauses and back, we describe an automatic framework integrating the two solving techniques and exchanging newly found information between them. Using examples derived from cryptographic attacks, we present some initial experiments indicating the efficiency of this combination.

Keywords: Boolean polynomial · Border Basis Algorithm
SAT solving · Cryptographic attack

1 Introduction

Cryptographic attacks frequently require the solution of polynomial systems defined over the field $\mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$ for which it is known that the desired solution consists of one or more points in $\mathbb{F}_2^n$. In this case we may add the *field equations* $x_i^2 + x_i = 0$ to the given system, where $i = 1, \ldots, n$, to express that fact that we are looking for solutions $(a_1, \ldots, a_n)$ such that $a_i \in \mathbb{F}_2$. Equivalently, we may consider the system as a system of Boolean polynomials, i.e., a system defined by elements in

$$\mathbb{F}_2[x_1, \ldots, x_n] \,/\, \langle x_1^2 + x_1,\ \ldots,\ x_n^2 + x_n \rangle.$$

Several methods have been developed to deal with this task.

Algebraic solvers consider the ideal I in $\mathbb{F}_2[x_1, \ldots, x_n]$ generated by the given polynomials and the field equations and perform operations such as polynomial addition and multiplication in order to find simple polynomials in I which allow us to read off the solutions of the system. Examples for such methods are Boolean Gröbner basis computations (see [3]) and the Boolean Border Basis Algorithm (BBBA) (see [11,12]). After converting the polynomial system to a set of propositional logic clauses (see [1,10,13]), one can also use a *SAT solver* to determine

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 147–162, 2017.
https://doi.org/10.1007/978-3-319-72453-9_11

a satisfying assignment for the logical variables, which in turn corresponds to a solution of the Boolean polynomial system (see for instance [5]). SAT solvers use logical reasoning (such as CDCL and DPLL) to eliminate large sets of assignments which do not satisfy the given set of clauses.

In this paper we combine both algebraic solvers and SAT solvers by running two processes in parallel and interchanging information between them. More specifically, we describe an implementation of an automatic framework which executes the Boolean Border Basis Algorithm (see [11]) and the SAT solver `antom` (see [16]) concurrently, transforms newly found "interesting" polynomials resp. SAT clauses using suitable conversion methods (see [10]), and introduces this new information into the other process. Up to now, Gröbner bases computations have been used predominantly to speed up certain stages of a SAT solver computation (see [6,7,17]). Hence this paper may be considered as a first step towards a systematic combination of algebraic and logical reasoning in order to solve Boolean polynomial systems.

Let us describe its contents in more detail. In Sect. 2 we recall some efficient algorithms for performing certain operations with order ideals of terms used afterwards, and in Sect. 3 we remind the reader of Boolean polynomials and spell out an explicit method for their linear reduction and interreduction. In Sect. 4 we present a version of the Boolean Border Basis Algorithm (originally presented in [11]) which is closer to the actual implementation and which allows us to introduce the integration with the SAT solver explicitly at suitable points of the calculation.

Section 5 contains the description of the integration of this version of the BBBA with a SAT solver. In particular, we have to select which polynomials and which clauses we send to the respective other solver, keeping the amount of transmitted data under control and providing that information which has the best chances to improve the overall solving speed. Then, in Sect. 6, we describe the design of the actual communication process between the two solvers. This entails finding suitable entry points for the new information as well as a queuing process for these data until a suitable point in time for the insertion is reached. Section 7 contains some observations about the necessary modifications to a standard SAT solver such as `antom` (cf. [16]).

Finally, in Sect. 8 we report some preliminary results about speed-ups of the overall solving time we could achieve. Both for a manual insertion of new information as well as for the automatic communication process described above, we found cases with substantial improvements of the total solving time. However, sometimes the combination of the two processes was slower, and the effects depend strongly on the chosen selection strategies for the transmitted information. Thus further experimentation using the new tools is needed to optimize the synergies which we can achieve.

Unless explicitly stated otherwise, we use the basic definitions and results in [11,15].

2 Algorithms for Basic Operations with Order Ideals

The set of **squarefree terms** in the indeterminates $x_1, \ldots, x_n$ is denoted by $\mathbb{S}^n$. We order terms in $\mathbb{S}^n$ by a *degree compatible term ordering* σ. An **order ideal**

is a factor-closed set of terms. Let $\mathcal{O}$ be an order ideal in $\mathbb{S}^n$. A set of terms $C = \{t_1, \ldots, t_k\} \subseteq \mathcal{O}$ is called a set of **cogenerators** of $\mathcal{O}$ (or we say that C cogenerates $\mathcal{O}$) if every term in $\mathcal{O}$ divides one of the terms $t_1, \ldots, t_k$. A set of cogenerators $\{t_1, \ldots, t_k\}$ is called **minimal** if no term t_i divides t_j for $j \neq i$. Thus order ideals are represented by their (unique) minimal set of cogenerators. For a set of terms $C \subseteq \mathbb{S}^n$, we denote by $\langle C \rangle_{\mathrm{OI}}$ the order ideal cogenerated by C.

In this section, three different non-trivial operations with order ideals used in the BBBA are discussed briefly. The order ideal membership problem is decided by Algorithm 1. Its proof of correctness follows immediately from the definition of cogenerators.

Algorithm 1. (Order Ideal Membership Test)
Input: Cogenerators C of an order ideal $\mathcal{O}$ in $\mathbb{S}^n$, $t \in \mathbb{S}^n$.
Output: `True` if $t \in \mathcal{O}$, `False` otherwise.

```
1: a := False
2: foreach c in C do
3:    if t divides c then
4:       a := True
5:    end if
6: end foreach
7: return a
```

The **squarefree border** of an order ideal $\mathcal{O}$ in $\mathbb{S}^n$ is defined as $\partial\mathcal{O}^{\mathrm{sf}} = \left(\left(\bigcup_{i=1}^{n} x_i\mathcal{O}\right) \setminus \mathcal{O}\right) \cap \mathbb{S}^n$. Next we present Algorithm 3 (and its subroutine Algorithm 2) for computing the minimal set of cogenerators of an order ideal minus a monomial ideal.

Algorithm 4 decides if the squarefree border of one order ideal is contained in some other order ideal. These algorithms are variants of Propositions 7.4 and 7.5 in [11]. Step 11 of Algorithm 2 and Step 12 of Algorithm 4 can be computed by removing terms that are divisible by others.

3 Linear Interreduction for Boolean Polynomials

In the following we let $\mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$ be the binary field and $\mathbb{F}_2[x_1, \ldots, x_n]$ a polynomial ring over $\mathbb{F}_2$. The ideal $F = \langle x_1^2 + x_1, \ldots, x_n^2 + x_n \rangle$ is called the **field ideal**. The ring $\mathbb{B}_n = \mathbb{F}_2[x_1, \ldots, x_n]/\langle F \rangle$ is called the **ring of Boolean polynomials** in the indeterminates $x_1, \ldots, x_n$. We assume that its elements are represented by polynomials whose support consists only of squarefree terms, i.e. all operations with polynomials are performed modulo the field ideal.

A set of Boolean polynomials $G \subseteq \mathbb{B}_n$ is called linearly LT_σ**-interreduced** if $\mathrm{LT}_\sigma(g) \neq \mathrm{LT}_\sigma(g')$ for all $g, g' \in G$ with $g \neq g'$. Given an arbitrary set of Boolean polynomials $G \subseteq \mathbb{B}_n$, we can linearly LT_σ-interreduce G via (sparse) Gaußian elimination on the coefficient matrix of G. (Here the columns have to be ordered w.r.t. σ.) For a better understanding of linear LT_σ-interreduction,

Algorithm 2. (Order Ideal Minus a Monomial Ideal Generated by a Term)
Input: $t \in \mathbb{S}^n$, the minimal set of cogenerators C of an order ideal $\mathcal{O}$ in $\mathbb{S}^n$.
Output: The minimal set of cogenerators of the order ideal $\mathcal{O} \setminus \langle t \rangle$.

$D := \emptyset$
foreach c **in** C **do**
  if t divides c **then**
    for $i = 1$ **to** n **do**
      if x_i divides t **then**
        $D := D \cup \{\frac{c}{x_i}\}$
      end if
    end for
  end if
end foreach
Let A be the set of the minimal elements in $(C \cup D) \setminus \langle t \rangle$ w.r.t. division.
return A

Algorithm 3. (Order Ideal Minus a Monomial Ideal)
Input: The minimal set of cogenerators C' of an order ideal U in $\mathbb{S}^n$, a set of squarefree terms T.
Output: The minimal set of cogenerators C of the order ideal $U \setminus \langle T \rangle$.
Requires: Algorithm 2.

$C := C'$
foreach t **in** T **do**
  if $t \in \langle C' \rangle_{\mathrm{OI}}$ **then**
    $C :=$Algorithm 2(t, C)
  end if
end foreach
return C

we formulate the following definitions which are analogous to the rewriting rules in the Gröbner basis theory (see [15, Definition 2.2.1]).

Definition 1. *Let $V \subseteq \mathbb{B}_n$, and let $b, r, b' \in \mathbb{B}_n$.*

(a) We say that b **linearly LT_σ-reduces to b' in one step** *using r if $\mathrm{LT}_\sigma(b) = \mathrm{LT}_\sigma(r)$ and $b' = a + r$. We write $b \xrightarrow{r} b'$.*

(b) We say that b **linearly LT_σ-reduces to** *b' using V if there exist $v_i \in V$ for $i = 1, \ldots, k$ and $b_1, \ldots, b_{k-1} \in \mathbb{B}_n$ such that $b \xrightarrow{v_1} b_1 \xrightarrow{v_2} \ldots \xrightarrow{v_{k-1}} b_{k-1} \xrightarrow{v_k} b'$. We write $b \xrightarrow{V} b'$.*

(c) A polynomial b with the property that there is no $r \in V$ such that $b \xrightarrow{r} b'$ for some $b' \in \mathbb{B}_n$ is called **linearly LT_σ-irreducible** *with respect to V.*

Obviously, we have $b \xrightarrow{b} 0$ for any polynomial b. The following example shows us that the result of a sequence of linear LT_σ-reductions is not uniquely determined in general.

Algorithm 4. (Checking the Border)
Input: Cogenerators C' of an order ideal U in $\mathbb{S}^n$, cogenerators D of an order ideal $\mathcal{O}$ in $\mathbb{S}^n$.
Output: `True` if $\partial\mathcal{O}^{\mathrm{sf}} \subseteq U$, `False` otherwise; a set of squarefree terms C such that $\langle C\rangle_{\mathrm{OI}} = \langle C' \cup \partial\mathcal{O}^{\mathrm{sf}}\rangle_{\mathrm{OI}}$.

1: $a :=$ `True`
2: $B := \emptyset$
3: **foreach** d **in** D **do**
4: **for** $i = 1$ **to** n **do**
5: $d' := x_i d$
6: **if** $d' \in \mathbb{S}^n$ **and** $d' \notin \langle C'\rangle_{\mathrm{OI}}$ **then**
7: $B := B \cup \{d'\}$
8: $a :=$ `False`
9: **end if**
10: **end for**
11: **end foreach**
12: Let C be the set of the minimal elements in $(C' \cup B)$ w.r.t. division.
13: **return** (a, C)

Example 1. Let $V = \{x_1x_2 + 1, x_1x_2, x_1 + 1\} \subseteq \mathbb{B}_2$. Then $x_1x_2 + x_1 \xrightarrow{x_1x_2+1} x_1 + 1 \xrightarrow{x_1+1} 0$, and thus $x_1x_2 + x_1 \xrightarrow{V} 0$. On the other hand, $x_1x_2 + x_1 \xrightarrow{x_1x_2} x_1 \xrightarrow{x_1+1} 1$, and thus $x_1x_2 + x_1 \xrightarrow{V} 1$.

If we would like to have unique linear LT_σ-reducers (and hence unique linear LT_σ-reductions), the set V has to be linearly LT_σ-interreduced.

Proposition 1. *Let V be a linearly LT_σ-interreduced set of Boolean polynomials. Let $b, b' \in \mathbb{B}_n$ such that $b \xrightarrow{V} b'$. Then the polynomial b' is uniquely determined.*

Proof. There exists exactly one element $v_1 \in V$ such that $\mathrm{LT}_\sigma(b) = \mathrm{LT}_\sigma(v_1)$, because the leading terms of the elements of V are pairwise distinct. Let $b_1 = b - v_1$. We have $b \xrightarrow{v_1} b_1$. There exists at most one element $v_2 \in V$ such that $\mathrm{LT}_\sigma(b_1) = \mathrm{LT}_\sigma(v_2)$. If there is no such v_2, the element b_1 is the unique linear LT_σ-reduction of b. Otherwise, we continue with $b_2 = b_1 - v_2$ in the same way, and the result follows by induction. □

The following proposition gives us another useful property of a linearly LT_σ-interreduced set of Boolean polynomials.

Proposition 2. *Let V be a linearly LT_σ-interreduced set of Boolean polynomials, and let $b, r \in \mathbb{B}_n$. Then we have $b \xrightarrow{V} 0$ if and only if $b \in \langle V\rangle_{\mathbb{F}_2}$.*

Proof. First we prove "⇒". By definition, there exist $v_1, \ldots, v_k \in V$ and $b_i \in \mathbb{B}_n$ for $i = 1, \ldots, k-1$ such that $b \xrightarrow{v_1} b_1 \xrightarrow{v_2} \ldots \xrightarrow{v_{k-1}} b_{k-1} \xrightarrow{v_k} 0$. Hence we get $b = v_1 + \cdots + v_k$ in $\mathbb{B}_n$, and henceforth $b \in \langle V\rangle_{\mathbb{F}_2}$.

Conversely, let $b = v_1 + \cdots + v_k$ for some pairwise distinct elements $v_1, \ldots, v_k$ in V. Because V is linearly LT_σ-interreduced, there exists a unique index $i_1 \in \mathbb{N}$ with $1 \leq i_1 \leq k$ such that $\mathrm{LT}_\sigma(v_{i_1}) = \mathrm{LT}_\sigma(b)$. Hence $b \xrightarrow{v_{i_1}} (b - v_{i_1})$. There exists a unique $i_2 \in \mathbb{N}$ with $1 \leq i_2 \leq k$ such that $\mathrm{LT}_\sigma(v_{i_2}) = \mathrm{LT}_\sigma(b - v_{i_1})$. By induction we create a zero linear LT_σ-reduction chain starting from b and having linear LT_σ-reducers $v_{i_1}, \ldots, v_{i_k} \in V$. □

Example 2. Let $V = \{x_1x_2 + x_1, x_1x_2 + x_2\} \subseteq \mathbb{B}_2$. We can see that $x_1 + x_2 \in \langle V \rangle_{\mathbb{F}_2}$, but $x_1 + x_2$ is linearly LT_σ-irreducible with respect to V.

We are now ready to introduce and analyze Algorithm 5 for computing successive extensions of linearly LT_σ-interreduced sets. As a pivoting strategy for the reduction process, we first consider Boolean polynomials of smallest degree and among them the ones having smallest support. Algorithm 5 will be applied in Algorithm 8 in the next section.

Definition 2. *Let $f, g \in \mathbb{B}_n$. We write $f \prec g$ if and only if* $\deg(f) < \deg(g)$, *or* $\deg(f) = \deg(g)$ *and* $\#\mathrm{Supp}(f) < \#\mathrm{Supp}(g)$.

Algorithm 5. (Extensions of Linearly LT_σ-Interreduced Tuples)
Input: A non-zero Boolean polynomial b', a linearly LT_σ-interreduced set of Boolean polynomials V', and a degree compatible term ordering σ.
Output: A set $V \subseteq \mathbb{B}_n$ such that V is linearly LT_σ-interreduced and $\langle V \rangle_{\mathbb{F}_2} = \langle V' \cup \{b'\} \rangle_{\mathbb{F}_2}$.

1: $b := b', V := V'$
2: **while** there exists $r \in V$ with $\mathrm{LT}_\sigma(r) = \mathrm{LT}_\sigma(b)$ **do**
3: $\quad b := b + r$
4: **end while**
5: **if** $b \neq 0$ **then**
6: $\quad V := V \cup \{b\}$
7: **end if**
8: **return** V

Proposition 3. *Algorithm 5 returns a linearly* LT_σ*-interreduced list V such that $\langle V \rangle_{\mathbb{F}_2} = \langle V' \cup \{b'\} \rangle_{\mathbb{F}_2}$ holds.*

Proof. In Step 2 we search for a unique polynomial in V' which has the same leading term as b. If such a polynomial does not exist, the polynomial b is appended to V in Step 6.

The linear LT_σ-reduction chain is constructed in Steps 2–4. If $b' \xrightarrow{V'} 0$, then $b' \in \langle V' \rangle_{\mathbb{F}_2} \subseteq \langle V \rangle_{\mathbb{F}_2}$ by Proposition 2. If we have $b' \xrightarrow{V'} b \neq 0$, then we have $b' \in V$ by Step 6. □

4 The Boolean Border Basis Algorithm

To start with, we recall the definition of a Boolean $\mathcal{O}$-border basis (see [11]).

Definition 3. *Let $P = \mathbb{F}_2[x_1, \ldots, x_n]$, let $\mathcal{O} = \{t_1, \ldots, t_\mu\}$ be an order ideal in $\mathbb{S}^n$, and let $\partial\mathcal{O}^{\mathrm{sf}} = \{b_1, \ldots, b_\nu\}$ be its squarefree border. Let $I \subseteq \mathbb{F}_2[x_1, \ldots, x_n]$ be an ideal containing the field ideal $F = \langle x_1^2 + x_1, \ldots, x_n^2 + x_n\rangle$.*

(a) A set of polynomials $G = \{g_1, \ldots, g_\nu\}$ is called a **Boolean $\mathcal{O}$-border prebasis** *if $g_j = b_j + \sum_{i=1}^{\mu} c_{ij}\, t_i$ with $c_{1j}, \ldots, c_{\mu j} \in \mathbb{F}_2$ for $j = 1, \ldots, \nu$.*

(b) A Boolean $\mathcal{O}$-border prebasis $G \subset I$ is called a **Boolean $\mathcal{O}$-border basis** *of I if the residue classes $\overline{\mathcal{O}} = \{\bar{t}_1, \ldots, \bar{t}_\mu\}$ in P/I form an $\mathbb{F}_2$-basis of P/I.*

Let us motivate the idea of the BBBA using the problem of finding the $\mathbb{F}_2$-rational solutions of a Boolean system $f_1 = \cdots = f_s = 0$. Let $V = \{f_1, \ldots, f_s\}$. Define the ideal $I = \langle f_1, \ldots, f_s\rangle \subseteq \mathbb{B}_n$. Suppose that the system has a unique $\mathbb{F}_2$-rational solution. (For instance, this is common in the scenario of algebraic attacks.) We are looking for a set of linear polynomials $G \subseteq I$ such that G is a linearly LT_σ-interreduced basis of $\langle G\rangle_{\mathbb{F}_2}$ and $\#\mathrm{Supp}(G) = \#G$. Hence the goal is to create new linearly independent linear polynomials in I and to keep the support of polynomials in the system as small as possible at the same time.

Given a set of Boolean polynomials $V = \{f_1, \ldots, f_s\}$, the BBBA generates new polynomials by forming and linearly LT_σ-interreducing $V^{(+)} = V \cup x_1 V \cup \cdots \cup x_n V$. Note that the multiplications are done in $\mathbb{B}_n$. Every iteration of $V^{(+)}$ is then followed by linear LT_σ-interreduction. One could repeat these two operations in order to obtain the desired basis. On the other hand, this approach clearly leads to an exponentially large amount of work since all polynomials in V are multiplied by n indeterminates.

Thus the operation $V^{(+)}$ in the BBBA is restricted by the order ideal U. The order ideal U is called the **universe** and U is initially cogeneratored by $\bigcup_i \mathrm{Supp}(f_i)$. The $V^{(+)}$ operation is restricted to polynomials that have their support contained in the universe. In this way, the growth of V and the support of the polynomials in V is lower. The universe is extended by the support of polynomials that have leading terms contained in U. This extension of the universe is described in Algorithm 6 which is used in Step 12 of Algorithm 8. The proof of correctness of Algorithm 6 is easy and left to the reader.

The successive computation of $V^{(+)}$ tends to repeat the consideration of multiples of polynomials that have been already multiplied by all indeterminates. To avoid this overhead, we introduce the following notion.

Definition 4. *A Boolean polynomial $f \in \mathbb{B}_n$ is said to be* **covered** *in a linearly LT_σ-interreduced set $V \subseteq \mathbb{B}_n$ if $x_i f \xrightarrow{V} 0$ for all $i \in \{1, \ldots, n\}$.*

Covered polynomials should be avoided because they do not introduce any new leading terms. The definition is equivalent to the condition $x_i f \in \langle V\rangle_{\mathbb{F}_2}$ for $i = 1, \ldots, n$ by Proposition 2. Checking the latter condition is quite expensive for large sets V. When we repeat the $V^{(+)}$ operation and linear LT_σ-interreduction, we remember the polynomials that have been worked on as in the following example.

Algorithm 6. (Extension of the Universe)
Input: Cogenerators C' of an order ideal U in $\mathbb{S}^n$, a linearly LT_σ-interreduced set of Boolean polynomials V.
Output: A set of cogenerators $C \supseteq C'$ such that $\mathrm{LT}_\sigma(f) \in \langle C \rangle_{OI}$ for $f \in V$ implies that f is contained in $\langle C \rangle_{OI}$.

1: $C := C'$
2: **repeat**
3: $D := C$
4: **foreach** f **in** V **do**
5: **if** $\mathrm{LT}_\sigma(f) \in \langle C \rangle_{\mathrm{OI}}$ **and** f is not contained in $\langle C \rangle_{OI}$ **then**
6: Let A be the set of the minimal cogenerators of $\langle C \cup \mathrm{Supp}(f) \rangle_{OI}$.
7: $C := A$
8: **end if**
9: **end foreach**
10: **until** $\#D = \#C$
11: **return** C

Example 3. Let $f = x_1x_2 + 1 \in \mathbb{B}_2$ and $V' = \{f\} \subseteq \mathbb{B}_2$. Let us compute $V'^{(+)}$ iteratively with successive linear LT_σ-interreduction. We compute $x_1 f = x_1x_2 + x_1 \xrightarrow{f} x_1+1$ and $x_2 f = x_1x_2+x_2 \xrightarrow{f} x_2+1$. We get $V = \{x_1x_2+1, x_1+1, x_2+1\}$. Then f is covered in V, and therefore multiplication of x_1x_2+1 by indeterminates does not yield new linearly independent polynomials during the computation of $V^{(+)}$. Thus we remember that the polynomial f is covered in V.

Algorithm 7 computes $\{b\}^{(+)}$ for b a Boolean polynomial and immediately linearly LT_σ-reduces the result against the known polynomials. To keep the pseudo-code simple, the covered polynomials that are easily discoverable are stored in the set $M \subseteq V$. The proof of correctness of Algorithm 7 follows directly from Proposition 3.

Now we describe a restructured version of the BBBA in Algorithm 8. Its subroutine `FinalReduction` refers to the algorithm in [14, Proposition 17] whose purpose is to extract the desired border basis from $\langle V \rangle_{\mathbb{F}_2}$. Notice that this algorithm can be easily modified to output only the polynomials having squarefree border terms.

Proposition 4. *In the setting of Algorithm 8, Algorithm 8 outputs the Boolean $\mathcal{O}_\sigma(I)$-border basis of I.*

Proof. It is sufficient to prove that Algorithm 8 is equivalent to Algorithm 4.3 in [11]. The set V_a denotes the set of all polynomials in V which are contained in the current universe $U = \langle C \rangle_{\mathrm{OI}}$. Note that V may contain polynomials which are not in $\langle C \rangle_{\mathrm{OI}}$. Thus the set V in Algorithm 4.3 in [11], corresponds to V_a.

The only difference in the initialization (apart from defining the new set M) occur in Steps 3–5. They are equivalent to linear LT_σ-interreducing of the initial generators V.

Algorithm 7. (Plus and Reduce)
Input: A non-zero Boolean polynomial b, a linearly LT_σ-interreduced set of Boolean polynomials V', a degree compatible term ordering σ, cogenerators C of an order ideal U in $\mathbb{S}^n$, and a set $M' \subseteq V$ of covered polynomials in V.
Output: A linearly LT_σ-interreduced set V such that $\langle V' \cup \{x_1 b, \ldots, x_n b\}\rangle_{\mathbb{F}_2} = \langle V\rangle_{\mathbb{F}_2}$ if b is contained in $\langle C\rangle_{\mathrm{OI}}$, $V = V'$ otherwise, and a set of covered polynomials M.
Requires: Algorithm 5.

1: $V := V'$, $M := M'$
2: **if** b is contained in $\langle C\rangle_{OI}$ and $b \notin M$ **then**
3: **for** $i = 1$ **to** n **do**
4: $b' := x_i b$
5: Update V by calling Algorithm 5(b', V, σ).
6: **end for**
7: $M := M \cup \{b\}$
8: **end if**
9: **return** (V, M)

Now we would like to show that Steps 7–13 computes the $\langle C\rangle_{\mathrm{OI}}$-stabilization of V_a, i.e. that $\langle V_a\rangle_{\mathbb{F}_2} = \langle V_a^{(+)}\rangle_{\mathbb{F}_2} \cap \langle U\rangle_{\mathbb{F}_2}$ holds in Step 14. The inclusion "$\subseteq$" is trivial. Let us look at the other inclusion. The set M contains polynomials in V such that $M^{(+)} \subseteq \langle V\rangle_{\mathbb{F}_2}$, so elements in M can be omitted in Algorithm 7.

Let $U = \langle C\rangle_{\mathrm{OI}}$ and $v \in \langle V_a^{(+)}\rangle_{\mathbb{F}_2} \cap \langle U\rangle_{\mathbb{F}_2}$ in Step 14. We know that $v \xrightarrow{V} 0$ because $\langle V_a^{(+)}\rangle_{\mathbb{F}_2} \subseteq \langle V\rangle_{\mathbb{F}_2}$ after Step 11. This means that $v \in \langle V\rangle_{\mathbb{F}_2}$ by Proposition 2 because V is linearly LT_σ-interreduced. We would like to show that $v \xrightarrow{V_a} 0$, which is equivalent to $v \in \langle V_a\rangle_{\mathbb{F}_2}$. Let $v = v_1 + \cdots + v_k$, where $\{v_1, \ldots, v_k\} \subseteq V$ is a linearly LT_σ-interreduced set. Then $\mathrm{LT}_\sigma(v) = \mathrm{LT}_\sigma(v_i)$ for some $1 \leq i \leq k$. Since $\mathrm{LT}_\sigma(v_i) = \mathrm{LT}_\sigma(v) \in U$, we get $v_i \in \langle U\rangle_{\mathbb{F}_2}$, i.e. $v_i \in V_a$ after Step 12. We continue with the polynomial $v - v_i$ and we get that $\{v_1, \ldots, v_k\} \subseteq V_a$ by induction.

The loop in Steps 9–11 enlarges V by elements in $\langle V_a^{(+)}\rangle_{\mathbb{F}_2}$ such that updated V is linearly LT_σ-interreduced. (This is equivalent to Step 5 of Algorithm 4.3 in [11].) Step 12 enlarges the universe in the same way as Steps 6–10 of Algorithm 4.3 in [11] do.

The rest (i.e., Steps 15–17) continues in the same way as Steps 13–16 of Algorithm 4.3 in [11]. □

5 The Integration of the BBBA with a SAT Solver

Many search problems can be encoded as systems of Boolean polynomials or SAT-instances. Inputs of SAT-solvers are usually in CNF (Conjunctive Normal Form), i.e. a conjunction of disjunctions of literals, where a literal is either a logical variable or its negation.

Algorithm 8. The BBBA (Restructured Version)
Input: A set of polynomials $V = \{f_1, \ldots, f_s\} \subseteq \mathbb{B}_n$ such that $V \cup F$ generates a 0-dimensional ideal I and a degree compatible term ordering σ.
Output: The polynomials of the Boolean $\mathcal{O}_\sigma(I)$-border basis of I where $\mathcal{O}_\sigma(I) = \mathbb{S}^n \setminus \mathrm{LT}_\sigma(I)$.
Requires: Algorithms 3, 4, 5, 6, 7, `FinalReduction`.

1: $V := \emptyset, M := \emptyset$
2: Let C be a set of the minimal cogenerators of the order ideal $\langle \bigcup_{i=1}^{s} \mathrm{Supp}(f_i) \rangle_{\mathrm{OI}}$.
3: **for** $i = 1$ **to** s **do**
4: Update V by calling Algorithm 5(f_i, V, σ).
5: **end for**
6: **repeat**
7: **repeat**
8: $V' := V$
9: **foreach** f chosen in the increasing order according to "$\prec$" **in** V' **do**
10: Update (V, M) by calling Algorithm 7(f, V, σ, C, M).
11: **end foreach**
12: $C :=$Algorithm 6(C, V).
13: **until** $\#V = \#V'$
14: $D :=$ Algorithm 3$(C, \mathrm{LT}_\sigma(V))$.
15: Update (a, C) by calling Algorithm 4(C, D).
16: **until** $a =$ `True`
17: Apply `FinalReduction`$(V, \langle D \rangle_{\mathrm{OI}})$ and return the result.

One can solve the same problem with the BBBA or a SAT solver individually. There exist conversion methods that transform a Boolean system to a CNF formula (and vice versa) such that the $\mathbb{F}_2$-rational zeros of the system correspond to the satisfying assignments of the logical formula. Thus we may run both solvers in parallel and let them interchange the "new information", or one solver can dynamically help the another one with a certain subproblem, etc. We will focus on the scenario when an algebraic solver helps a SAT solver because it provides the best results according to our initial experiments. For more details on conversions, see [10].

Previously, we had handled the interaction of two solvers manually. During our experiments, several examples were observed where one solver is sped up by utilizing information derived by the other. Based on these observations, the communication was automated with a view towards optimizing the achievable gains.

The integration is tailored to be applicable for most SAT solvers. For our experiments, we used the SAT solver `antom` [16]. Modern SAT solvers are mainly based on CDCL. They produce many *conflict clauses* which contain new information that can be potentially used in the BBBA after a conversion. On the other hand, any new polynomial found in the ideal by the BBBA can be converted and sent to a SAT solver. To reduce the amount of information that needs to be transferred, we transmit only short clauses and short polynomials of a low

degree. Moreover, an additional filtering technique has been developed to further reduce the number of clauses that are handled by the BBBA. This selection strategy makes the BBBA sufficiently fast to keep up with the SAT solver. In this way the BBBA is not stuck with computations which are potentially outdated and irrelevant for the SAT solver by the time they are finished.

Description of the integration. Assume that the SAT solver is running on a given CNF in the background. Our approach is divided into 7 steps (viewed from the BBBA side) which repeat until the SAT solver stops:

1. *Receiving clauses.* The SAT solver sends a set of new conflict clauses C that it has generated to the BBBA.
2. *Clause filtration.* We define a subset $C' \subseteq C$, where C' contains clauses $c \in C$ such that there exist $c' \in C$ with $c \neq c'$ that shares at least one variable with c. We buffer only the first 10 clauses on an as-they-come basis.
3. *Converting clauses to polynomials.* We use the standard conversion [10, Algorithm 1] to produce Boolean polynomials from the selected clauses.
4. *Computing a border basis.* We call Algorithm 8 on the output of the previous step. We restrict the sets of indeterminates of the Boolean ring to the indeterminates actually appearing in the input polynomials. We do not apply `FinalReduction`.
5. *Polynomial filtration.* We choose only linear, quadratic or cubic polynomials produced by Algorithm 8 that are different from the input of the BBBA. Among them, we select polynomials with the smallest support.
6. *Converting polynomials to clauses.* We convert these polynomials to clauses via the (sparse) truth-table method described in [10, Example 1]. We buffer only the first 100 clauses on an as-they-come basis.
7. *Sending the clauses.* We send these clauses to the SAT solver and go to Step 1.

6 Design of the Communication

To combine the power of the SAT solver with the advanced reasoning of the BBBA, a severe communication challenge has to be overcome. While it would be possible to create a fully integrated BBBA-SAT hybrid solver, the maintenance of such a solver would be difficult and the implementation of new features into either base solver challenging. Therefore, a communication framework that allows the exchange of data between the border basis and the SAT solver is developed instead. The design of this communication layer focused on two objectives: 1. The overhead for the data transfer must be low. 2. The base solvers should be modified as little as possible.

To achieve the first design goal, a shared memory communication approach is chosen. By defining a shared memory region that is accessible to both solvers, large amounts of data can be transmitted at extremely high speeds. To satisfy the second objective, the communication is restricted to consist only of clauses. Furthermore, the shared memory communication is implemented with the help of the `Boost Interprocess` Library [9]. This library allows different

processes to access a common, shared memory region. Thus, the SAT solver and the BBBA can be executed independently and simply access the same shared memory region.

The communication itself is combined into a handler class that performs the generation of the shared memory region and the coordination and synchronization of the data access. It furthermore provides a simple interface for the sending and receiving of clauses. The handler class only needs to be instantiated by the solvers to gain access to the shared memory region.

When the BBBA and the SAT solver are combined, there are two instances of the shared memory manager that are communicating. Apart from the initialization of the shared memory region that is performed at the start of the application, these instances behave exactly in the same way. For simplicity, the two managers are referred to as m_1 and m_2 here. To allow for an efficient communication, each manager utilizes its own shared memory area for outgoing clauses, o_1 and o_2. This enables a fast full duplex communication, as each manager can send and receive at the same time.

When m_1 is asked to transmit a clause to m_2, it first stores the clause in a local queue. The next clause of the local queue is transferred to the shared region o_1 when m_2 indicates that it read the previously shared clause from that area. Once the clause has been written to o_1, m_1 informs m_2 that a new clause is available. The manager m_2 then copies the clause from o_1 to its own memory region and marks the clause as read. Thus, the next clause in the queue of m_1 can be transmitted.

To avoid any idle waiting in the background, the check for new clauses and the transmission of the next clause are only performed when the solvers update their shared memory handler.

7 Modifications of the SAT Solver

The SAT solver constantly generates new conflict clauses. The shear volume of conflict clauses makes it unfeasible to share all of them with the BBBA. Instead, only conflict clauses below a certain size threshold are transmitted. A new conflict clause is transmitted immediately after it has been generated. This modification adds only a single line of code to the solver.

Receiving new clauses is slightly more challenging because of the way clauses are stored and considered in the solver `antom` [16]. For efficiency reasons, the first literal of every clause that is not satisfied must be free (i.e., currently not assigned to a value). Therefore, each new clause that is received from the BBBA is first sorted and then added. Depending on the variable assignment that is currently under consideration by the SAT solver, a new clause might, furthermore, be unsatisfied at the moment. In this case, the solver backtracks until the clause is not unsatisfied anymore. Here the new clause acts similar to a conflict clause and guides the solver away from unsatisfied regions of the search space. The additional tasks that are required to handle a received clause are placed into a new function. Thus, the main SAT solver code only needs to be extended by a single line of code that checks for the arrival of new clauses.

The shared memory handler is updated once after every decision. This is sufficiently often to receive any new clauses, but does not add any undue overhead to the solver.

Overall, the SAT solver is modified only to a very small extent. Hence the solver can be freely developed without worrying about complex dependencies. Similarly, it should be comparatively easy to add the presented communication layer to a different SAT solver, should the need arise.

8 Experiments and Timings

To evaluate the combination of BBBA and `antom`, two different kinds of experiments have been performed. Timings in this paper were obtained on a computer under Linux having a 2.60 GHz Intel Core i7-5600U CPU and a total of 16 GB RAM. We note here that `antom` is deterministic, i.e. it gives the same result on the same input for each run.

8.1 Manual Combination of the Information

The first set of experiments is meant to showcase the general usefulness of the information that is derived by the BBBA for the SAT solver. Let C be a CNF input instance for `antom`. We convert C into a set of Boolean polynomials S via the standard conversion. Next we run the BBBA for 5 min and then stop the execution. We select one linear polynomial f in V manually and convert f back to CNF via the truth-table method. Let C' be its result. We run `antom` twice: once with the input C and then with the input $C \wedge C'$. The timings in Table 1 illustrates the speed-up obtained by manual section of extra information provided by the BBBA.

Table 1. Comparison of timings of `antom` on the Small Scale AES instances in [8] without vs with extra clauses corresponding to a linear polynomial.

CNF instances	antom	antom + lin. poly
AES-2-1-2-8	11.80	3.50
AES-2-2-1-8	111.59	88.15
AES-2-2-4-4	196.06	21.76
AES-1-2-4-8	666.93	209.11
AES-2-4-2-4	3997.91	1432.24

8.2 Automatic Combination of the Information

For the second set of experiments, the newly developed Algorithm 8 and the integration framework with `antom` in C++ as described in Sect. 5 were used. In Table 2 we present the timings of this automation on various benchmarks. Instances

`factoring`x, y were generated by [2]. They encodes the factoring problem for $x \cdot y$. The other benchmarks encode algebraic attacks or algebraic fault attacks on the cryptosystems Small Scale AES and LED-64. For the full description of these benchmarks, we refer to [4,8]. The timeout limit was set to 1200 s.

Table 2. Timings of the integration of the BBBA with `antom` vs vanilla `antom` for various SAT instances.

CNF instances	antom	BBBA + antom
factoring81551,100057	0.23	0.22
AES-2-2-4faultInNibble1with1faultyBits	3.12	2.35
factoring3981643,3981641	7.83	6.91
factoring2190823,2190821	19.53	74.25
factoring7367627,7367621	29.18	146.13
factoring12619463,12619427	40.43	101.34
AES-4-4-4faultInNibble1with4faultyBits	41.15	55.87
LED64faultInNibble1with1faultyBits	45.70	55.38
AES-4-4-4faultInNibble1with1faultyBits	49.02	48.61
factoring5160011,5160007	63.98	55.47
factoring5621809,5621809	81.54	110.07
factoring4752977,4752949	207.18	189.58
factoring5308571,5308553	282.91	37.22
AES-2-2-4-4algebraicCNF	268.70	235.39
factoring49987277,49999553	337.58	45.29
factoring12598967,12598951	441.88	78.60
factoring4593761,4593737	527.22	10.11
factoring5287813,5287801	605.76	102.48
factoring5620907,5620907	653.63	5.04
factoring10000079,10000019	>1200	760.41

During our experiments we found examples where the integration was slower than the SAT solver by itself. In practice, we therefore suggest to run the SAT solver alone on one machine and the integration in parallel on another machine. In this way, we cannot be "unlucky" and we will always profit from the best timings. This is particularly relevant in cryptanalytic scenarios where the solution of an instance implies breaking a cryptosystem.

Notice that the timings of the integration of the two solvers are sometimes not stable, i.e. two timings for the same instance may differ substantially. These differences occur because new clauses are added at different points in time.

The filtration techniques described in Sect. 5, as well as the integration itself, are still preliminary. Our next goal is to develop deeper understanding of the synergy of both solvers. The main difficulty is that SAT solvers use various heuristics

for literal assignment and for the choice on what clause to work on next. This makes it very hard to analyze which extra clauses from the BBBA affect the timings most. Nonetheless, our results show that the additional information from the BBBA already greatly increases the speed of the SAT solver.

Acknowledgments. We would like to express our gratitude to Tobias Schubert for providing us with the source code of the SAT solver `antom`. This work was financially supported by the DFG project "Algebraische Fehlerangriffe" [KR 1907/6-1].

References

1. Bard, G., Courtois, N., Jefferson, C.: Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over GF(2) via SAT-solvers. In: IACR Cryptology ePrint Archive (2007). https://eprint.iacr.org/2007/024.pdf
2. Bebel, J., Yuen, H.: Hard SAT instances based on factoring. In: SAT Competition 2013: Solver and Benchmark Descriptions, University of Helsinki, p. 102 (2013)
3. Brickenstein, M.: Boolean Gröbner Bases: Theory, Algorithms and Applications. Logos Verlag, Berlin (2010)
4. Burchard, J., Gay, M., Messeng Ekossono, A.S., Horáček, J., Becker, B., Schubert, T., Kreuzer, M., Polian, I.: AutoFault: towards automatic construction of algebraic fault attacks. In: Proceedings of Conference on Fault Diagnosis and Tolarence in Cryptography (FDTC 2017), Taipei (2017, to appear)
5. Burchard, J., Messeng Ekosonno, A.-S., Horáček, J., Gay, M., Becker, B., Schubert, T., Kreuzer, M., Polian, I.: Towards mixed structural-functional models for algebraic fault attacks on ciphers. In: Proceedings of International Verification and Security Workshop (IVSW 2017) (2017)
6. Condrat, C., Kalla, P.: A Gröbner basis approach to CNF-formulae preprocessing. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 618–631. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71209-1_48
7. Dreyer, A., Nguyen, T.H.: Improving Gröbner-based clause learning for SAT solving industrial sized Boolean problems. In: Young Researcher Symposium (YRS) (Kaiserslautern 2013), Fraunhofer ITWM, pp. 72–77 (2013)
8. Gay, M., Burchard, J., Horáček, J., Messeng Ekossono, A.S., Schubert, T., Becker, B., Kreuzer, M., Polian, I.: Small scale AES toolbox: algebraic and propositional formulas, circuit-implementations and fault equations. In: Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2016), Barcelona (2016)
9. Gaztanaga, I.: The Boost Interprocess Library, version 1.63.0. www.boost.org/doc/libs/1_63_0/doc/html/interprocess.html
10. Horáček, J., Kreuzer, M.: On conversions from CNF to ANF. In: 2th International Workshop on Satisfiability Checking and Symbolic Computation, SC-square, Kaiserslautern (2017)
11. Horáček, J., Kreuzer, M., Messeng Ekossono, A.S.: Computing Boolean border bases. In: Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2016), Timisoara, pp. 465–472. IEEE (2016)
12. Horáček, J., Kreuzer, M., Messeng Ekossono, A.S.: A signature based border basis algorithm. In: Conference on Algebraic Informatics, CAI, Kalamata (2017)

13. Jovanovic, P., Kreuzer, M.: Algebraic attacks using SAT-solvers. Groups Complex. Cryptol. **2**, 247–259 (2010)
14. Kehrein, A., Kreuzer, M.: Computing border bases. J. Pure Appl. Algebra **205**(2), 279–295 (2006)
15. Kreuzer, M., Robbiano, L.: Computational Commutative Algebra 1. Springer, Heidelberg (2000)
16. Schubert, T., Reimer, S.: Antom (2016). https://projects.informatik.uni-freiburg.de/projects/antom
17. Zengler, C., Küchlin, W.: Extending clause learning of SAT solvers with boolean Gröbner bases. In: Gerdt, V.P., Koepf, W., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2010. LNCS, vol. 6244, pp. 293–302. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15274-0_26

Isabelle Formalization of Set Theoretic Structures and Set Comprehensions

Cezary Kaliszyk[1] and Karol Pąk[2(✉)]

[1] Universität Innsbruck, Innsbruck, Austria
cezary.kaliszyk@uibk.ac.at
[2] Uniwersytet w Białymstoku, Białystok, Poland
pakkarol@uwb.edu.pl

Abstract. Reasoning about computers and programming languages on paper is most often done with set theory, while most proof assistant formalizations of languages and programs use alternative mathematical foundations. One of the few exceptions has been Mizar where the *Simple Concrete Model* of computers has been used to verify programs expressed as abstract programming language instruction sequences. The model uses extended set theory features including structures and Fraenkel set comprehension operators. In this paper we show how to formally specify such objects in the Isabelle object logic implementing the Mizar foundations as definitional extensions. To show the adequacy and usability of the mechanisms, we reformalize a number of Mizar definitions and theorems related to structures and set comprehensions, including both mathematical and programming language examples: groups, machines and properties of computer memory states.

Keywords: Isabelle · Mizar · Structure · Set comprehension
Multiple inheritance

1 Introduction

Proof assistants are today increasingly used to certify software, hardware, as well as mathematical proofs that involve computer programs [10]. One of the earliest proof assistants, Mizar [7], has been developed as a tool to provide a human-oriented environment which would allow proofs to be formally analyzed. The system has already been developed over forty years with its most distinctive features being a proof style that imitates informal mathematical proofs as much as possible [16] and a rich type system that reflects how mathematicians and computer scientists describe dependencies between objects [25]. Such support for formal proofs has given rise to one of the largest libraries of formalized mathematics with many domains not covered in other libraries. One of such

The paper has been supported by the resources of the Polish National Science Center granted by decision n°DEC-2015/19/D/ST6/01473.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 163–178, 2017.
https://doi.org/10.1007/978-3-319-72453-9_12

domains is the *Simple Concrete Model* (SCM) [17], which introduces a formal model corresponding to random-access Turing machines, their instructions, and programs and has been considered more realistic for modeling of real computers [21]. The development of SCMs and the proofs of their various properties spans 66 *Mizar Mathematical Library* (MML) articles.

We build upon our recent work which aimed to specify foundations [13], notations [12], automation [14] of the Mizar system in the Isabelle Logical Framework [24]. The most important motivation for the current work is to provide the completely specified set theoretic formalizations of the model of computers, instructions, languages, etc. There are multiple further directions for how we plan to extend that work, as well as multiple reasons for these extensions:

- Specifying Mizar in a logical framework gives the complete semantics of the system specified only on paper so far [6], including the underlying first-order logic variant, the soft type system, definitional mechanisms, and automation mechanisms.
- Despite various efforts [11], the contents of the MML are hard to access for developers of other proof and knowledge management systems. Isabelle's structures can allow experiments with sharing proof techniques and automation across proof assistants.
- Mizar has a large monolithic kernel. Despite the implementers best efforts, bugs in the code can result in incorrect proofs being accepted. This problem can be significantly remedied by certifying proofs across systems.
- In the long run, develop an alternative environment for reverification and development of proofs automatically exported from the MML.

In this work we introduce and develop two components used in Mizar necessary to translate and certify the MML proofs on algebraic structures including the SCM model of computers in Isabelle. The components are Mizar *structures* and Mizar *set comprehension* operators.

Mizar *structures* (also referred to as *aggregates* or *records*) allow grouping multiple other objects together with relations between them into a single entity. This is useful for defining and reasoning about mathematical structures such as rings, fields, and vector spaces. Mizar structures correspond to mechanisms in other proof assistants like the Isabelle type classes [9] or Coq records used to build an algebraic hierarchy [5]. The support for structures is a crucial part of the Mizar language. Structures are built in to the Mizar VERIFIER [6] and they are heavily used in the MML. In fact 74% of the articles in the current MML version 1289 rely directly or indirectly on the article struct_0.

Mizar *set comprehension* operators (referred to as *Fraenkel* in the Mizar literature [6]) allow describing a set of terms whose argument list satisfies a given predicate. Defining set comprehension in a sound and adequate way is an important part of the Mizar foundations, as in any set theory this is where most paradoxes (Russell's paradox and its variants) originate from.

We use the already specified foundations of the Mizar system together with the Tarski-Groethendieck axiomatization, and the first few formalized articles

of the MML to formally define Mizar structures and Mizar set comprehension operators. Both can be introduced as definitional extensions, without adding any further axioms.

1.1 Related Work

B-Method [1] has aimed to ease the formalization of programs in a foundation based on set theory, however like Mizar the structures and set comprehension are a part of the system. Similarly, Metamath [20] does not have a built-in notion of structures and focuses on n-tuples instead.

Turing machines have been formalized in Isabelle/HOL [26] allowing reasoning about their behavior in Hoare logic, as well as in Matita [2] focusing on complexity theory. The main approach to formalization of imperative programs in Isabelle is used in Imperative/HOL [4]. This approach was further refined to allow for formalization of programs in separation logic [18].

Algebraic structures were often necessary early in the development of proof assistants. In Isabelle/HOL type classes [9] allow for further control of the polymorphic type system adding mechanisms such as inheritance between types. Various Isabelle automation mechanisms can translate type classes to predicates, which is also how reasoning about algebraic structures with inheritance is usually performed in other HOL-based systems. Proof assistants based on versions of type theory can store objects along with their properties in tuples (or records with named fields). This has been used to build an algebraic hierarchy [5] in Coq or to extend it to topologies as done in Matita [23]. Inheritance for records that can allow for good automation has become an important field with developments including canonical structures and type classes.

Lee and Rudnicki [19] proposed an alternative approach to defining structures without special support in the Mizar system. The main motivation is to make field structures into first-class objects, which allows more convenient reasoning about graphs. The proposed approach directly uses other parts of the Mizar language (including preceding parts of the MML) to define aggregates as Mizar `finite Functions`. This allows defining what it means for an object to have a field, rather than to fix which collections of fields constitute an aggregated object.

The exports of Mizar to ATPs [3] require a specification of the Mizar set comprehensions. The semantics of the exported objects is the same as that in Mizar and in our formalizations, but they are axiomatized rather than defined. We are not aware of any work that specifies the foundations of Mizar in a formal system that would cover structures.

1.2 Contributions and Outline

We give a complete formal specification of Mizar structures formalized in the Isabelle/Mizar object logic (Sect. 3). It supports strict structures (structures that

do not include additional fields), domain of a structure (which allows restricting larger structures to smaller ones), and inheritance (which allows extending structures to larger ones) including multiple inheritance.

We formally specify the Mizar Fraenkel set comprehension operator (Sect. 4). Our approach allows defining it as a single meta-level functor, therefore a definitional extension as opposed to a part of the implementation of the CHECKER in Mizar. We further prove a number of properties of this functor.

We reformalize parts of the MML corresponding to the lattice of types focusing on the *simple concrete model* of computers, and show that the defined mechanisms are appropriate and usable to formalize all of Mizar specifics in Isabelle (Sect. 5).

2 Preliminaries

In this section we briefly introduce the Mizar foundations defined as an object logic in the Isabelle framework. For more details see [13].

Four Isabelle types are used to model the Mizar foundations. The type of propositions is already defined by the underlying Isabelle/FOL object logic. The following types are further added: the type of Mizar sets `Set` and two types used for the Mizar type system: `Mode` and `Attr`. Mizar *modes* are the elementary types assigned to all objects. Modes are guaranteed to be non-empty. Mizar *attributes* allow restricting of a given mode or of another attribute. The only attributes considered in this paper will be *adjectives*. Each adjective corresponds to a (parameterizable) predicate on a given type. The type constructed by applying a number of adjectives to a given type corresponds to the elements of the type which satisfy all the adjective-associated predicates. For example the type `non zero natural number` restricts the type (mode) of numbers to both natural ones and those different from zero. For clarity, in the Isabelle formalization the operation that combines attributes will be denoted using a single vertical bar | and the operation of applying attributes to a mode will be denoted using a double one ‖. More information about modes and attributes can be found in [7].

The Isabelle/Mizar object logic introduces constants that allow interacting with the Mizar types, a constant for the choice operator, and five axioms that specify these constants. Two axioms specify what it mean to define a new mode and a new attribute. Two axioms express the meaning of the combinations of attributes with attributes and with modes. The last one axiomatizes the Mizar axiom of choice for non-empty types.

Next, notations that imitate the Mizar text are introduced for the first-order logic symbols: `&`, `or`, `implies`, `for x holds P`, etc. Syntax and helper lemmas are provided to allow defining functions, predicates, and new types in ways similar to that used in Mizar. In particular the definition of a meta-level function `F` which is to return type `T` in Mizar follows the pattern `func F → T equals D` and definitions using the description operator use `means` rather than `equals` and a predicate that the defined object should satisfy. These preliminaries are sufficient to express the Tarski-Grothendieck set theory axiomatization in the

same way as in Mizar. Furthermore [13] showed, that it is sufficient to translate all the definitions and theorems from the first few articles of the MML.

3 Structures

Mizar structures are used to define objects that are typically represented as tuples in mathematics. For example the Mizar definition of ring $\langle F, +, 0, \cdot, 1 \rangle$ consists of Mizar types assigned to fields in the structure, in particular $+$, $\cdot$ are binary operations on F, and 0, 1 are members of F. To do this, unique identifiers (referred to as a *field selector* or simply *selector* in the Mizar literature) are needed for each tuple element. In case of a ring these identifiers are `carrier`, `addF`, `ZeroF`, `multF`, and `OneF` respectively. The Mizar syntax for the tuple including the above mentioned types is presented on the left. The Isabelle counterpart, which we will define later in this section is presented on the right for comparison (for simplicity inheritance information is omitted here, it will be discussed in Sect. 3.4):

```
struct doubleLoopStr (#
   carrier → set,
   addF → BinOp of the carrier,
   ZeroF → Element of the carrier,
   multF → BinOp of the carrier,
   OneF → Element of the carrier
#)
```

```
definition
"struct doubleLoopStr (#
   carrier → λS. set;
   addF → λS. BinOp-of the carrier of S;
   ZeroF → λS. Element-of the carrier of S;
   multF → λS. BinOp-of the carrier of S;
   OneF → λS. Element-of the carrier of S
#)"
```

The `doubleLoopStr` structure will correspond to a ring only with additional restrictions. Such restrictions are in Mizar introduced using adjectives (see Sect. 2). In particular, a ring in the MML is defined as a `doubleLoopStr` together with nine adjectives, such as `Abelian` and `distributive` with their expected meanings. Certain extensions of a ring, such as a field, will only extend the list of adjectives (for example by `commutative`), which permits all Mizar mechanisms (functors, definitions, theorems) associated with rings to also work with fields. Mizar allows adjectives to be used in the field selector types, which corresponds to structures with restricted values. This is used for SCMs (see Sect. 5.2).

Mizar structures also support inheritance discussed in more detail in Sect. 3.4. Here it is only important to note that inheritance does allow not only ring extensions, but also permits the use of group theory for rings and fields, since the group tuple `multLoopStr` is a sub-tuple of that of `doubleLoopStr`. This means that "being a group" defined for `multLoopStr` must allow tuples that have more than the required selectors. However, there are cases where we want to express the fact that a group has precisely the `multLoopStr` selectors, namely that the tuple does not have any other elements. This is achieved using the Mizar attribute strict that can be applied to any structure, which specifies that only the selectors from that structure are allowed. The need for strict can be illustrated

by the following example. Consider the set of all groups over $\mathbb{Z}_3$. This set is finite if and only if we consider strict structures. The net hierarchy of basic algebraic structures in the MML is depicted in Fig. 1.

3.1 Structure Preliminaries

In the Mizar literature the word structure is used both for *structure prototypes* (e.g. the type of rings) and for actual structure instances (e.g. individual objects that are of the type of rings). We will try to distinguish the two when it is not clear from the context. Structure instances will be represented as set theoretic functions. We will use our Isabelle reformalization of the Mizar set theoretic relations for this purpose. Structure prototype definitions will correspond to schemes of functions, which can be further restricted by the given adjectives.

3.2 Structure Operations

A structure prototype definition will describe functions given as sets of assignments. Each assignment is of the form $x \rightarrow y$, where x is a unique label (selector) and y is the specification given to that field of the structure. As the specification may refer to the other parts of the structure (for example the zero in the ring is an element of the carrier), y needs to be a meta-level function which, when given the structure instance as an argument returns the type of that field. We present here the general definitions of the selector and of the single field in a structure in our formalization:

```
definition TheSelectorOf ("the _ of _ " [90,90] 190) where
   "func the selector of Term → object means λit.
      for T be object st ⟨selector,T⟩ in Term holds it = T"

definition Field ("_ → _" 91) where
   "selector → spec ≡ define_attr (λit.
      the selector of it be spec(it) & selector in dom it)"
```

With this we can introduce a Mizar-like syntax for structure prototypes $(\#\,\mathtt{f}_1;\ldots;\mathtt{f}_n\,\#)$, where each field f_i is described by an assignment $sel \rightarrow spec(it)$. Most basic structure prototypes ignore the argument:

```
definition one_sorted :: "Mode"  ("one-sorted") where
  "struct one-sorted (# carrier → λ_. set #)"
```

We now define the domain of a structure prototype as the minimal set that is contained in the domain of any instance. This allows the following definition to be a global one, however the result makes sense only for a particular prototype.

```
definition domain_of::"Mode ⇒ Set" ("domain'_of _" 200) where
  "func domain_of M → set means (λit.
      (ex X be M st it = dom X) & (for X be M holds it ⊆ dom X))"
```

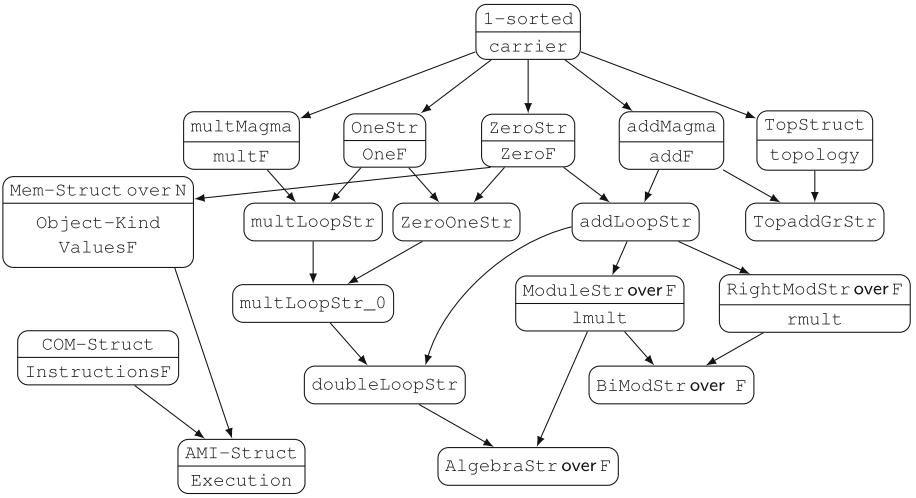


Fig. 1. Net of basic algebraic and computer-related structures in the MML following [8]. The presented ones have already been covered in our formalization. The lower part of each node lists the selectors which are added w.r.t. the inherited ones. *AMI* (Architecture Model for Instructions) is an abstract computer structure parametrized by the data stored in its memory, further detailed in Sect. 5.2.

The fact that we know the domain globally also allows creating strict as an attribute. The attribute should consider the domain of the structure type, which may be the last argument after other attributes. Not to restrict the order of attributes, the Isabelle version of strict requires an argument, which repeats the mode. For example strict one-sorted || one-sorted.

```
definition strict :: "Mode ⇒ Attr" ("strict _" 200) where
  "attr strict M means (λX. X be M & dom X = domain_of M)"
```

We can finally introduce the restriction of an instance to a strict structure using the restriction of a function domain denoted with the slash operator.

```
definition the_restriction_of :: "Set ⇒ Mode ⇒ Set"
                  ("the'_restriction'_of _ to _" 90) where
  "func the_restriction_of X to Struct →
     strict Struct ‖ Struct equals X | domain_of Struct"
```

3.3 Structure Prototype Introduction

To define an actual structure prototype, it is necessary to use an actual set of labels which are pairwise different. In principle strings could be natural for this purpose. However, as we prefer to reduce the required part of the library foundations, we chose to use the set theoretic natural numbers defined by $0 = \{\}$ and $succ(X) = X \cup \{X\}$.

Furthermore, to define an actual structure prototype, it is necessary to show non-emptiness, that is that there exists a structure instance which fulfills the structure prototype conditions. To show such existence, Mizar requires the non-emptiness of all structure field specifications. For this, we use the global choice operator ϵ. For each field (selector $\rightarrow$ specification) we take the pair $\langle$selector, ϵ(specification)$\rangle$. We show that the set of such pairs for all fields of the structure fulfills the prototype conditions (with convenient automation to show such existence, see `Mizar_struct` file). For example, for `doubleLoopStr` we use:

```
term "{⟨carrier, the set⟩} ∪
  {⟨addF, the BinOp-of the set⟩} ∪ {⟨ZeroF, the Element-of the set⟩} ∪
  {⟨multF, the BinOp-of the set⟩} ∪ {⟨OneF,the Element-of the set⟩}"
```

3.4 Inheritance and Multiple Inheritance

The original MML definition of `doubleLoopStr` includes information about (multiple) inheritance:

struct (addLoopStr,multLoopStr_0) doubleLoopStr

which informs Mizar that `doubleLoopStr` should inherit all the fields contained in `addLoopStr`, as well as those in `multLoopStr_0`. These are used to define additive and multiplicative groups, respectively. Inheritance is transitive. Multiple inheritance causes the Mizar inheritance graph to become a DAG. There are 168 structures defined in MML. This does not include their versions with adjectives. Most structures (135 of them) inherit from `1-sorted`. A part of the graph restricted to the most basic algebraic structures is depicted in Fig. 1. In our approach it is possible to verify that the domain of a structure is a subset of the domain of another one allowing (automated) inheritance proofs at any point after the definition.

4 Set Comprehension

Set comprehension is a key notion in Mizar set theory. It allows defining a set of terms, which satisfy the given predicate (see [6, Fraenkel]), with Mizar syntax:

$$\{\mathtt{t}(\mathtt{v}_1,\mathtt{v}_2,\ldots,\mathtt{v}_n) \text{ where } \mathtt{v}_1 \text{ is } \Theta_1,\ \mathtt{v}_2 \text{ is } \Theta_2,\ldots,\mathtt{v}_n \text{ is } \Theta_n : \mathtt{P}[\mathtt{v}_1,\mathtt{v}_2,\ldots,\mathtt{v}_n]\}$$

Such an expression is of the type `set` in Mizar and it is not possible to further specify the type. The built-in definition of the set comprehension operator is automatically expanded in terms of set membership as follows:

$$\mathtt{x} \text{ in } \{\mathtt{t}(\mathtt{v}_1,\mathtt{v}_2,\ldots,\mathtt{v}_n) \text{ where } \mathtt{v}_1 \text{ is } \Theta_1,\ \mathtt{v}_2 \text{ is } \Theta_2,\ldots,\mathtt{v}_n \text{ is } \Theta_n : \mathtt{P}(\mathtt{v}_1,\mathtt{v}_2,\ldots,\mathtt{v}_n)\}$$
$$\text{iff}$$
$$\text{ex } \mathtt{v}_1 \text{ be } \Theta_1,\ \mathtt{v}_2 \text{ be } \Theta_2,\ \ldots,\mathtt{v}_n \text{ be } \Theta_n \text{ st } \mathtt{x} = \mathtt{t}(\mathtt{v}_1,\mathtt{v}_2,\ldots,\mathtt{v}_n) \ \&\ \mathtt{P}[\mathtt{v}_1,\mathtt{v}_2,\ldots,\mathtt{v}_n]$$

The generality of the definition could quickly lead to a version of the Russell's paradox, as according to the Tarski-Groethendieck axiomatization everything is a set. Therefore, the set comprehension operator is well-formed only when all the types $\Theta_1, \Theta_2, \ldots, \Theta_n$ have the sethood property (otherwise Mizar reports Error 86: *"It is only meaningful for sethood property"*, see [7] for more details).

Definition 1. *A Mizar-type Θ has the sethood property if all objects of the type Θ are elements of some set.*

We define sethood in Isabelle/Mizar and make sure that it is proved for the most important Mizar types (Mizar allows the inheritance of sethood). With this, we can show the existence of sets described by comprehensions. The Isabelle/Mizar statement and proof are quite involved, so we present these mostly in mathematical setting.

Theorem 1. *Let Θ be a Mizar type with the sethood property, P be a unary predicate and F be a unary function defined on Θ. Then there exists a set C such that each x is a member of C if and only if there exists a v of type Θ such that $x = F(v) \wedge P(v)$.*

Proof. The proof only relies on the Tarski-Groethendieck axiom of Replacement. Consider the set S_{ethood} that contains all objects of the type Θ. Furthermore, consider the binary relation R_1 defined for a predicate P as $R_1(x, y) \iff x = y \wedge P(x)$. Then, by the axiom of Replacement, there exists a set $S_{eparation}$, such that x is a member of $S_{eparation}$ if and only if there exists y that is a member of S_{ethood} and $R_1(x, y)$. Now $S_{eparation}$ contains the objects of the type Θ which satisfy P and only such objects. We can use the Replacement axiom again for the unary relation $\lambda y.\exists x.\, y = F(x)$ and the set S_{ethood}. This gives the image of the function F on the set S_{ethood}. This set fulfills the requirement of the theorem statement. □

The theorem was so far limited to unary predicates and functions. To adapt it to multiple arguments, we can consider the Cartesian product together with the property that two tuples are equal, if their corresponding elements are equal. In our Isabelle/Mizar formalization we introduced the Cartesian product in the zfmisc_1 theory corresponding to the Mizar article with the same name. This can used to show set comprehensions with multiple arguments:

Theorem 2. *Let $\Theta_1, \Theta_2, \ldots, \Theta_n$ be Mizar types with the sethood property, P be an n-argument predicate and F be an n-argument function defined for the arguments of the types $\Theta_1, \Theta_2, \ldots, \Theta_n$. Then there exists a set C such that x is a member of C if and only if there exists v_1 be Θ_1, v_2 be $\Theta_2, \ldots, v_n$ be Θ_n, such that $x = F(v_1, v_2, \ldots, v_n) \wedge P(v_1, v_2, \ldots, v_n)$.*

Proof. Consider the sets S_i which contain objects of the types Θ_i. Consider the binary relation R_1, defined as

$$\lambda xy.x = y \wedge \exists_{v_1, v_2, \ldots, v_n} x = \langle\langle \ldots \langle\langle v_1, v_2\rangle, v_3\rangle, \ldots\rangle, v_n\rangle \wedge P(v_1, v_2, \ldots, v_n)$$

The Replacement axiom can be used to obtain the set $S_{eparation}$ for which x is a member of $S_{eparation}$ if and only if there exists y that is a member of $(\ldots((S_1 \times S_2) \times S_3)\ldots) \times S_n$ and $R_1(x, y)$. Using the Replacement axiom again for the relation

$$\lambda xy.\exists_{v_1,v_2,\ldots,v_n} x = \langle\langle\ldots\langle\langle v_1, v_2\rangle, v_3\rangle,\ldots\rangle, v_n\rangle \wedge y = F(v_1, v_2, \ldots, v_n)$$

and the set S_{ethood}, we obtain the set C. □

The following example collects the results of the function `f` on the set `X` and can be shown to be equivalent to the range of the function restricted to the set.

```
term "{f. x where x be Element-of dom f: x in X}"
```

Set comprehensions are often used in the MML to define sets of terms without additional properties. The following syntax has been introduced so simplify such comprehension terms: the set of all t(v1,v2,...,vn) where v1 is Θ1, v2 is Θ2,...,vn is Θn which abbreviates: {t(v1,v2,...,vn) where v1 is Θ1, v2 is Θ2,...,vn is Θ2: non contradiction}. Just like for set comprehensions we add this abbreviation together with the Mizar notation. It can be seen for example in the following theorem:s

```
theorem funct_1_th_110:
  assumes "B be non empty | functional || set"
          "f be Function" "f = union B"
  shows
     "dom f = union the set-of-all dom g where g be Element-of B"
     "rng f = union the set-of-all rng g where g be Element-of B"
```

5 Case Studies

In this section we argue that our model of structures is not only correct based on the Tarski-Groethendieck set theory axioms, but also that it is adequate for Mizar-like formalization. For this, we formalized a part of Mizar's group theory in Isabelle defining the basic concepts as structures, the corresponding attributes, and showing a number of their properties. We also show how groups combine with set comprehensions and a more involved inheritance example. All Isabelle examples have same identifiers as their Mizar counterparts to ease comparison.

5.1 Algebraic Structures

We first define the Mizar type of groups as the multiplicative magma structure `multMagma` with three adjectives. We define the identity in the group and an inverse, where the group operation is defined as usual.

```
abbreviation Group where
   "Group ≡ Group-like | associative | non empty-struct || multMagma"
```

```
definition group_1_def_4 ("1'._" [1000] 99) where
  "assume G is unital
   func 1._G → Element-of-struct G means λit.
      for h being Element-of-struct G holds
         h ⊗_G it = h & it ⊗_G h = h"

definition group_1_def_5 (infix "⁻¹„ 105) where
  "func h⁻¹_G → Element-of-struct G means λit.
      h ⊗_G it = 1._G & it ⊗_G h = 1._G"

definition algstr_0_def_18 ("_ ⊗_ _" [96, 1000, 97] 96) where
  "func x ⊗_M y → Element-of-struct M equals
     (the multF of M) . ⟨| x , y |⟩"
```

Next, we show a number of theorems about groups. We show here only a property that each group fulfils properties of semigroups with involution. The Mizar formalization does not need to repeat the variable declarations, thanks to the reserve mechanism, which is similar to Isabelle locales, but for each theorem only the variables and assumptions that are actually needed for its statement are exported. We do not have a complete mechanism of this kind yet.

```
theorem group_1_th_16:
  assumes "G be Group"
          "h be Element-of-struct G" "g be Element-of-struct G"
  shows "(h ⊗_G g)⁻¹_G = g⁻¹_G ⊗_G h⁻¹_G"
```

We have also reproved the 13 schemes that talk about set comprehension. We show here two, one that combines set comprehension with functions, and one that uses nested comprehensions.

```
theorem Fraenkel_sch_9:
  assumes "A be set" "B be set" "X be set"
          "f be Function-of A,B" "g be Function-of A,B"
          "(f | X) = (g | X)"
        "for u being Element-of A st u in X holds P(u) iff Q(u)"
  shows "{ f. u where u be Element-of A : P(u) & u in X } =
         { g. v where v be Element-of A : Q(v) & v in X }"

theorem Fraenkel_sch_13:
  assumes T0: "A be set" "B be set" "C be set"
   "for x1 be object,x2 be object holds F(x1,x2) be Element-of C"
  shows "{ st1 where st1 be Element-of C:
             st1 in {F(s1,t1) where s1 be Element-of A,
               t1 be Element-of B: P(s1,t1) } & Q(st1)} =
         { F(s2,t2) where s2 be Element-of A,t2 be Element-of B:
             P(s2,t2) & Q(F(s2,t2))}"
```

We finally look at the combination of groups and set comprehensions. The following two definitions introduce the set of all inverses and the set of results of the group operation:

```
definition group_2_def_1 (infix "~⁻¹ „ 150) where
  "func A~⁻¹_G → Subset-of-struct G equals
     {g⁻¹_G where g be Element-of-struct G : g in A}"
```

```
definition group_2_def_2(" _ ⊗_ _" [66, 1000, 67] 66) where
  "func A ⊗_G B → Subset-of-struct G equals
     {a ⊗_G b where a be Element-of-struct G,
                   b be Element-of-struct G : a in A & b in B}"
```

We can now show the relationship between these two operations, which is a consequence of the properties of semigroups with involution (`group_1_th_16` above).

```
theorem group_2_th_11:
  assumes "G be Group"
          "A be Subset-of-struct G" "B be Subset-of-struct G"
  shows "(A ⊗_G B) ~^-1_G = B~^-1_G ⊗_G A~^-1_G"
```

We finally show a multiple inheritance relation for the double loop structure. It follows by simple rewriting just using the definitions of the structures.

```
theorem doubleLoopStr_inheritance:
  assumes "X be doubleLoopStr"
  shows "X be multLoopStr_0" "X be addLoopStr"
```

5.2 SCM Computer Model

The MML models computers as structures whose elements correspond to: the set of instructions, the computer memory, and the functor `Execution` whose role is to map each instruction to a function from memory states to memory states.

The instructions form a set which must fulfill four properties corresponding to the following adjectives

```
abbreviation
  "Instructions ≡ J/A-independent|homogeneous|with_halt|standard-ins||set"
```

The `standard-ins` adjective specifies that any element `i` of the instruction set is a triple

```
term "[InsCode i, JumpPart i, AddressPart i]"
```

where `InsCode i` is an instruction number represented by a natural number, `JumpPart i` is a list of natural numbers used by the `Execution` functor to compute the following instruction numbers, and `AddressPart i` is a list objects passed to the instructions as arguments. The `with_halt` adjective means that the set `I` includes a halt instruction. The halt instruction is represented as `[0,{},{}]`, where the empty set corresponds to the empty list. The adjectives `homogeneous` and `J/A-independent` specify a subset of instructions which share the number `InsCode` and are necessary for the definition of the `Execution` functor. `homogeneous` specifies, that the `JumpPart` lists of arguments given to the `InsCode` instruction are always of the same length (for example `goto` always requires one argument). `J/A-independent` specifies that every list of the appropriate length can be handled (for the `goto` example, `Execution` must be able to perform a `goto` instruction to every location).

```
definition compos_0_def_5 ("homogeneous") where
  "attr homogeneous means (λI.
     I be non empty|standard-ins||set &
      (for i,j be Element-of I st InsCode i = InsCode j holds
         dom JumpPart i = dom JumpPart j))"

definition compos_0_def_7 ("J|A-independent") where
  "attr J|A-independent means (λI.
     I be non empty|standard-ins||set &
     (for n be Nat, f1,f2 be NAT-valued ||Function, p be object
        st dom f1 = dom f2 & [n,f1,p] in I holds [n,f2,p] in I))"
```

The next structure in the formalization is the computer memory (see Fig. 1). It is also modeled as a structure. The main field, `carrier`, corresponds to the actual memory and the set `N` gives the kind of data that can be stored within it. Note that that in SCMs all memory locations are of the same size [17]. The `ZeroF` field is the instruction counter. It corresponds to the number of the instruction performed in the given state. `Object-Kind` indicates the kind of data stored in the given memory location and `ValuesF` gives the value range for the given type.

```
definition MemStruct_over ("Mem-Struct-over _") where
"struct Mem-Struct-over N (#
   carrier → λS. set;
   ZeroF → λS. Element-of the carrier of S;
   Object-Kind → λS. Function-of the carrier of S, N;
   ValuesF →  λS. ManySortedSet-of N
#)"
```

An actual memory state is defined as a function that associates each memory location in the `carrier` with the stored data, where the value must be one of the allowed values.

```
definition memstr_0_def_2 ( "the'_Values'_of _ " 190) where
  "func the_Values_of M → ManySortedSet-of the carrier of M equals
     the ValuesF of M ∘ the Object-Kind of M"

abbreviation memstr_0_mode_2 ("State-of _" 190)
  where "State-of M ≡
        (the carrier of M):total | the_Values_of M-compatible || Function"
```

We can now formulate the Mizar type of a computer and show the non-emptiness of this type. The type is a structure parametrized by the data stored in the memory of the computer. The name AMI (for *Architecture Model for Instructions*) is used in the Mizar dictionaries to refer to this structure as a type and SCM is an object of the type [17].

```
definition AMI_Struct_over ("AMI-Struct-over _") where
  "struct AMI-Struct-over N (#
     carrier → λS. set;
     ZeroF → λS. Element-of the carrier of S;
     InstructionsF → λ_. Instructions;
     Object-Kind → λS. Function-of the carrier of S, N;
     ValuesF →  λS. ManySortedSet-of N;
     Execution → λS. Action-of the InstructionsF of S,
          product ((the ValuesF of S)*'the Object-Kind of S)#)"
```

We subsequently reformalize a machine with the halt instruction and we show that all the indicated fields have their corresponding types, and that this construction uniquely defines a computer. The proof corresponding to the below definition requires 83 lines of Isabelle proof to justify.

```
definition extpro_1_def_1 ("Trivial-AMI _") where
  "func Trivial-AMI N → strict AMI-Struct-over N‖AMI-Struct-over N
     means (λit.
      the carrier of it = {0} &
        the ZeroF of it = 0 &
the InstructionsF of it = {[0,{},{}]} &
  the Object-Kind of it = {0} --> 0  &
      the ValuesF of it = N --> NAT &
    the Execution of it = {[0,{},{}]} --> id product(N --> NAT ∘ {0} --> 0))"
```

Next, we introduce the `Exec` functor. Applying the instruction `I` to `(the Execution of S)` we should obtain a function that can be given a memory state as input and returns a memory state. Again showing the correctness of the definitions and that these properties hold requires 57 lines of Isabelle proofs.

```
definition extpro_1_def_2( "Exec _'(_ ,  _')" 190) where
  "func Exec _S(I,s) → State-of S equals
    ((the Execution of S).I).s "

definition extpro_1_def_3("halting _") where
  "attr halting S means (λI.
        I be Instruction-of S &
        (for s be State-of S holds  Exec _S(I,s) = s))"
```

We finally show, that `Trivial-AMI N` is of the computer type and that it does halt, which shows the non-emptiness of the Mizar type of computers.

```
theorem extpro_1:
  assumes "N be with_zero‖set"
  shows "halt_Trivial-AMI N is halting Trivial-AMI N"
```

6 Conclusion

Mizar structures and set comprehension operators complete the foundations of Mizar as an Isabelle object logic. This allows manual translation of the MML to Isabelle/Mizar, as we have shown with the Mizar theory of basic algebraic structures including SCMs. We have defined 90 concepts where 27 of them required justifications and proved 105 registrations, 31 theorems that discuss based algebraic structures and set comprehensions, as well as inheritance relations between 15 structures. We have defined also 27 concepts where 14 of them required justification and proved 12 registrations, 3 theorems about SCMs. The total combined size of the development is 513 kB and 8295 lines of proofs. It is available at: http://cl-informatik.uibk.ac.at/cek/macis2017/

The Isabelle proofs are mostly longer are than their Mizar counterparts. This is predominantly because of the lack of type automation for the type system, even if the Mizar type system could be handled by ATPs [15]. Similarly, many Isabelle proofs require more labels than the corresponding Mizar ones, which

we hope to remedy by developing legibility tools similar to the ones available for Mizar [22]. Finally it would be interesting to mechanically translate MML statements or even proofs and imitate the behavior of Mizar's automation.

Acknowledgements. This work has been supported by the European Research Council (ERC) grant no. 714034 *SMART* and the Polish National Science Center granted by decision n°DEC-2015/19/D/ST6/01473.

References

1. Abrial, J.: Modeling in Event-B - System and Software Engineering. Cambridge University Press, Cambridge (2010)
2. Asperti, A., Ricciotti, W.: A formalization of multi-tape turing machines. Theor. Comput. Sci. **603**, 23–42 (2015)
3. Brown, C.E., Urban, J.: Extracting higher-order goals from the Mizar mathematical library. In: Kohlhase, M., Johansson, M., Miller, B., de de Moura, L., Tompa, F. (eds.) CICM 2016. LNCS (LNAI), vol. 9791, pp. 99–114. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42547-4_8
4. Bulwahn, L., Krauss, A., Haftmann, F., Erkök, L., Matthews, J.: Imperative functional programming with Isabelle/HOL. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 134–149. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71067-7_14
5. Geuvers, H., Pollack, R., Wiedijk, F., Zwanenburg, J.: A constructive algebraic hierarchy in Coq. J. Symb. Comput. **34**(4), 271–286 (2002)
6. Grabowski, A., Korniłowicz, A., Naumowicz, A.: Mizar in a nutshell. J. Formaliz. Reason. **3**(2), 153–245 (2010)
7. Grabowski, A., Korniłowicz, A., Naumowicz, A.: Four decades of Mizar. J. Autom. Reason. **55**(3), 191–198 (2015)
8. Grabowski, A., Korniłowicz, A., Schwarzweller, C.: On algebraic hierarchies in mathematical repository of Mizar. In: Ganzha, M., Maciaszek, L.A., Paprzycki, M. (eds.) Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS 2016), pp. 363–371 (2016)
9. Haftmann, F., Wenzel, M.: Constructive type classes in Isabelle. In: Altenkirch, T., McBride, C. (eds.) TYPES 2006. LNCS, vol. 4502, pp. 160–174. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74464-1_11
10. Harrison, J., Urban, J., Wiedijk, F.: History of interactive theorem proving. In: Siekmann, J.H. (ed.) Computational Logic, Handbook of the History of Logic, vol. 9, pp. 135–214. Elsevier, Amsterdam (2014)
11. Iancu, M., Kohlhase, M., Rabe, F., Urban, J.: The Mizar mathematical library in OMDoc: translation and applications. J. Autom. Reason. **50**(2), 191–202 (2013)
12. Kaliszyk, C., Pąk, K.: Presentation and manipulation of Mizar properties in an Isabelle object logic. In: Geuvers, H., England, M., Hasan, O., Rabe, F., Teschke, O. (eds.) CICM 2017. LNCS (LNAI), vol. 10383, pp. 193–207. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62075-6_14
13. Kaliszyk, C., Pąk, K., Urban, J.: Towards a Mizar environment for Isabelle: foundations and language. In: Avigad, J., Chlipala, A. (eds.) Proceedings of the 5th Conference on Certified Programs and Proofs (CPP 2016), pp. 58–65. ACM (2016)

14. Kaliszyk, C., Pąk, K.: Progress in the independent certification of Mizar mathematical library in Isabelle. In: Ganzha, M., Maciaszek, L.A., Paprzycki, M. (eds.) Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS 2017), pp. 227–236 (2017)
15. Kaliszyk, C., Urban, J.: MizAR 40 for Mizar 40. J. Autom. Reason. **55**(3), 245–256 (2015)
16. Kaliszyk, C., Wiedijk, F.: Merging procedural and declarative proof. In: Berardi, S., Damiani, F., de'Liguoro, U. (eds.) TYPES 2008. LNCS, vol. 5497, pp. 203–219. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02444-3_13
17. Korniłowicz, A., Schwarzweller, C.: Computers and algorithms in Mizar. Mech. Math. Appl. **4**(1), 43–50 (2005)
18. Lammich, P.: Refinement to imperative/HOL. In: Urban, C., Zhang, X. (eds.) ITP 2015. LNCS, vol. 9236, pp. 253–269. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22102-1_17
19. Lee, G., Rudnicki, P.: Alternative aggregates in Mizar. In: Kauers, M., Kerber, M., Miner, R., Windsteiger, W. (eds.) Calculemus/MKM -2007. LNCS (LNAI), vol. 4573, pp. 327–341. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73086-6_26
20. Megill, N.D.: Metamath: A Computer Language for Pure Mathematics. Lulu Press, Morrisville (2007)
21. Nakamura, Y., Trybulec, A.: A mathematical model of CPU. Formaliz. Math. **3**(2), 151–160 (1992)
22. Pąk, K.: Automated improving of proof legibility in the Mizar system. In: Watt, S.M., Davenport, J.H., Sexton, A.P., Sojka, P., Urban, J. (eds.) CICM 2014. LNCS (LNAI), vol. 8543, pp. 373–387. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08434-3_27
23. Sacerdoti-Coen, C., Tassi, E.: Formalising overlap algebras in Matita. Math. Struct. Comput. Sci. **21**(4), 763–793 (2011)
24. Wenzel, M., Paulson, L.C., Nipkow, T.: The Isabelle framework. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 33–38. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71067-7_7
25. Wiedijk, F.: Mizar's soft type system. In: Schneider, K., Brandt, J. (eds.) TPHOLs 2007. LNCS, vol. 4732, pp. 383–399. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74591-4_28
26. Xu, J., Zhang, X., Urban, C.: Mechanising turing machines and computability theory in Isabelle/HOL. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) ITP 2013. LNCS, vol. 7998, pp. 147–162. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39634-2_13

Jordan Canonical Form with Parameters from Frobenius Form with Parameters

Robert M. Corless, Marc Moreno Maza, and Steven E. Thornton(✉)

ORCCA, University of Western Ontario, London, ON, Canada
sthornt7@uwo.ca

Abstract. The Jordan canonical form (JCF) of a square matrix is a foundational tool in matrix analysis. If the matrix A is known exactly, symbolic computation of the JCF is possible though expensive. When the matrix contains parameters, exact computation requires either a potentially very expensive case discussion, significant expression swell or both. For this reason, no current computer algebra system (CAS) of which we are aware will compute a case discussion for the JCF of a matrix $A(\alpha)$ where α is a (vector of) parameter(s). This problem is extremely difficult in general, even though the JCF is encountered early in most curricula.

In this paper we make some progress towards a practical solution. We base our computation of the JCF of $A(\alpha)$ on the theory of regular chains and present an implementation built on the `RegularChains` library of the Maple CAS. Our algorithm takes as input a matrix in Frobenius (rational) canonical form where the entries are (multivariate) polynomials in the parameter(s). We do not solve the problem in full generality, but our approach is useful for solving some examples of interest.

Keywords: Jordan form · Rational canonical form
Parametric linear algebra · Regular chains
Triangular decomposition

1 Introduction

The Jordan canonical form (JCF) of a matrix and its close cousin the Weyr canonical form are foundational tools in the analysis of eigenvalue problems and dynamical systems. For a summary of theory, see for instance Chap. 6 in The Handbook of Linear Algebra [1]; for the Weyr form, see [2].

The first use usually seen for the JCF is as a canonical form for matrix similarity: two matrices are similar if and only if they have identical (sets of, up to ordering) Jordan canonical forms [3]. Of course, there are other (often better) canonical forms for similarity such as the Frobenius (rational) canonical form, or the rational Jordan form [4,5].

This work is supported by The Natural Sciences and Engineering Research Council of Canada (NSERC).

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 179–194, 2017.
https://doi.org/10.1007/978-3-319-72453-9_13

The JCF is well known to be discontinuous with respect to changes in the entries if the base field $\mathbb{K}$ has nonempty open sets. We typically take $\mathbb{K} = \mathbb{C}$, the field of complex numbers. Therefore, the JCF cannot be computed numerically with small forward error, even when using a numerically stable algorithm. This has forced the development of alternatives to the JCF, such as the Schur form, which is numerically stable and useful in the computation of matrix functions via the Parlett recurrences, for instance [6]. Consider the computation of the matrix exponential. First computing the JCF is one of the famous "Nineteen Dubious Ways to Compute the Exponential of a Matrix" [6,7]; computing the matrix exponential remains of serious interest today (or perhaps is even of increased interest) because of new methods for "geometric" numerical integration of large systems [8–10].

Analysis of small systems containing symbolic parameters is also of great interest, in mathematical biology especially (models of disease dynamics in populations and in individual hosts, evolutionary or ecological models) but also in many other dynamical systems applications such as fluid-structure interactions, robot kinematics, and electrical networks. The algorithmic situation for systems containing parameters is much less well-developed than is the corresponding situation for numerical systems. Although alternatives to the JCF exist for the analysis of these systems, the JCF has become a standard tool with implementations available in every major CAS.

The current situation in Maple is that explicit computation of the JCF of a matrix containing parameters of dimension 5 or more may fail in some simple cases. For example, Maple simply does not provide a result for the JCF of the Frobenius companion matrix of $p(x) = x^5 + x^4 + x^3 + x^2 + x + a$. Similar failures occur for the `MatrixFunction` and `MatrixExponential` procedures. Wolfram Alpha gives the generic answers, but fails to give non-generic ones. Computing matrix functions may succeed in cases where computing the JCF does not because the JCF need not be used (an interpolation algorithm can be used instead); see for instance Definition 1.4 in [6].

Most computer algebra systems (CAS) have adopted some variation of the definition of algebraic functions as implicit roots of their defining polynomials. In Maple, the syntax uses `RootOf`; together with an `alias` facility. This gives a useful way to encode the mathematical statement (for instance) "Let α be a root of the polynomial $x^5 + \varepsilon x + 1 = 0$".

```
> alias(alpha = RootOf(x^5+eps*x+1,x)):
```

This should, in theory, allow symbolic computation of the JCF of (small) matrices, even ones containing parameters. To date in practice it has not.

In this paper, we offer some progress, although we note that combinatorial growth of the resulting expressions remains a difficulty. However, the tools we provide here are already useful for some example applications and go some way towards filling a scientific and engineering need. We aim to minimize unnecessary growth throughout the computation. The tools we use here include provisos [11] and comprehensive square-free factorization with the `RegularChains` package.

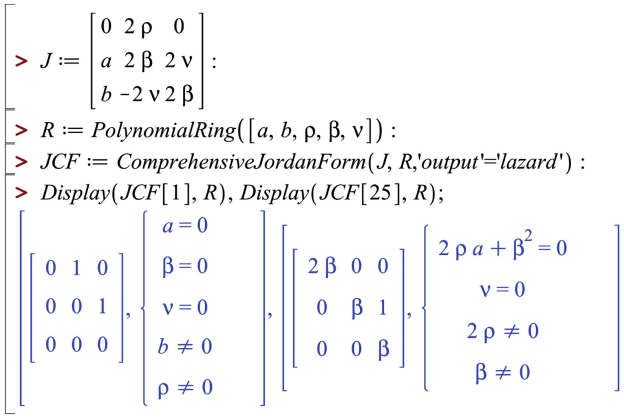


Fig. 1. Our implementation provides a full case discussion of the JCF of a matrix with 5 parameters. Two non-trivial cases are shown.

Consider, for example, the Jacobian matrix in [12]. Maple's built-in `JordanForm` command returns a diagonal matrix where the eigenvalues are large nested radical expressions as a result of explicitly solving the characteristic polynomial. In contrast, our `ComprehensiveJordanForm` method gives a full case discussion. Two interesting cases where the JCF is non-trivial are shown in Fig. 1. Further details of this example are given in Sect. 6.4.

In Sect. 5, we present an algorithm for computing the JCF of a matrix in Frobenius form where the entries are multivariate polynomials whose indeterminants are regarded as parameters. Our approach uses comprehensive square-free factorization to provide a complete case discussion. Classical approaches for computing the JCF rely on elementary row and column operations that maintain a similarity relation at each step [13–15]. Because the entries of the matrices we are considering are multivariate polynomials, row and column operations lead to significant expression growth that can be difficult to control. Additionally, this would require us to work over matrices of multivariate rational functions in the parameters, again making it difficult to control expression growth. By instead computing fraction free square-free factorizations, we are able to maintain better control over expression growth. Because our implementation does not use elementary row and column operations, we do not compute the similarity transformation matrix Q such that $J = Q^{-1}AQ$ gives the Jordan form J of A. We leave this problem for future work.

We present an implementation of our algorithm in Sect. 6 and use it to solve several problems taken from the literature. These examples are not in Frobenius form and we do not discuss in detail how we obtain the Frobenius form. The Frobenius form implementation uses standard algorithms based on GCD computations of parametric polynomials to find the Smith form of $A - xI$ and the relation between this and the Frobenius form of A [5].

Section 4 presents a new approach for computing the JCF of a non-parametric matrix in Frobenius form over the splitting field of the characteristic polynomial. Our discussion is based on the theory of regular chains. We do not apply this splitting field approach in the parametric case because the square-free factorization approach we use gives the complete structure of the JCF. Constructing the splitting field would be vastly more expensive than the approach of Sect. 5.

2 Some Prior Work

As previously mentioned, the JCF of a matrix $A \in \mathbb{C}^{n\times n}$ as a function of the entries of A has discontinuities. These discontinuities are often precisely what is important in applications. This also means that even numerically stable algorithms can sometimes give results with $\mathcal{O}(1)$ forward error. This is often also stated by saying that "computing the JCF is an ill-posed problem" [13]. This has not prevented people from trying to compute the JCF numerically anyway (see [13] and the references therein), but in general such efforts cannot always be satisfactory: discontinuous is ill-posed, and without regularization such efforts are (sometimes) doomed. There have been at least three responses in the literature.

One is to find other ways to solve your problem, i.e. compute matrix functions such as A^n and $\exp(tA)$, without first computing the JCF, and the invention of the numerically stable Schur factoring and the Parlett recurrences for instance has allowed significant success [6].

The second response is to find a canonical form that explicitly preserves the continuity or smoothness of the matrix; the versal forms of [16] do this. Incidentally, the Frobenius form with parameters is an example of a versal form (Arnol'd calls this a Sylvester family), but there are others. The paper [17] uses Carleman linearization to do something similar.

The third response is to assume exact input and try to do exact or symbolic computation of the JCF. Early attempts, e.g. [14], had high complexity: $\mathcal{O}(n^8)$ [15] in the dimension n and with expression growth $\mathcal{O}(2^{n^2})$. A key step is the computation of the Frobenius form, and the current best complexity algorithm is $\mathcal{O}(n^3)$ field operations, and keeps expression swell to a minimum [18]. Boolean circuit complexity results can be found in [5].

Inclusion of symbolic parameters makes things much more complicated and expensive, of course. Early work by Guoting Chen, who used with a single parameter [19] does not seem to have been improved upon. Some modern computer algebra systems simply give up when asked to compute the JCF of a matrix bigger than 5×5 that contains a parameter as we showed in Sect. 1.

There has been a significant body of relevant computational algebraic work, in computing the Frobenius form, the Zigzag form, and the Smith form [18,20] but relatively few works [16,19] on matrices with parameters. The difficulty appears to be combinatorial growth in the number of possible different cases. In the context of solving parametric linear systems, not eigenvalues, a significant amount of work has been done [21–26]. Parametric nonlinear systems are studied in [27–29] and the references therein.

3 Preliminaries

Sections 3.1 and 3.2 gather the basic concepts and results from polynomial algebra that are needed in this paper. Meanwhile, Sects. 3.3 and 3.4 review the notions of the Frobenius canonical form and the Jordan canonical form.

3.1 Regular Chain Theory

Let $\mathbb{K}$ be a field and $\overline{\mathbb{K}}$ its algebraic closure. Let $X_1 < \cdots < X_s$ be $s \geq 1$ ordered variables. We denote by $\mathbb{K}[X]$ the ring of polynomials in the variables $X = X_1, \ldots, X_s$ with coefficients in $\mathbb{K}$. For $F \subset \mathbb{K}[X]$, we denote by $\langle F \rangle$ and $V(F)$, the ideal generated by F in $\mathbb{K}[X]$ and the algebraic set of $\overline{\mathbb{K}}^s$ consisting of the common roots of the polynomials of F. For a non-constant polynomial $p \in \mathbb{K}[X]$, the greatest variable of p is called the *main variable* of p and denoted $\mathrm{mvar}(p)$, and the leading coefficient of p w.r.t. $\mathrm{mvar}(p)$ is called the *initial* of p, denoted by $\mathrm{init}(p)$. The Zariski closure of $W \subseteq \overline{\mathbb{K}}^s$, denoted by $\overline{W}$, is the intersection of all algebraic sets $V \subseteq \overline{\mathbb{K}}^s$ such that $W \subseteq V$ holds.

A set $T \subset \mathbb{K}[X] \setminus \mathbb{K}$ is *triangular* if $\mathrm{mvar}(t) \neq \mathrm{mvar}(t')$ holds for all $t \neq t'$ in T. Let h_T be the product of the initials of the polynomials in T. We denote by $\mathrm{sat}(T)$ the *saturated ideal* of T; if T is empty, then $\mathrm{sat}(T)$ is defined as the trivial ideal $\langle 0 \rangle$, otherwise it is the ideal $\langle T \rangle : h_T^\infty$. The *quasi-component* $W(T)$ of T is defined as $V(T) \setminus V(h_T)$. The following property holds: $\overline{W(T)} = V(\mathrm{sat}(T))$.

A triangular set $T \subset \mathbb{K}[X]$ is a *regular chain* if either T is empty, or the set T' is a regular chain, and the initial of p is regular (that is, neither zero nor zero divisor) modulo $\mathrm{sat}(T')$, where p is the polynomial of T with largest main variable, and $T' := T \setminus \{p\}$. Let $T \subset \mathbb{K}[X]$ be a regular chain. If T contains s polynomials $t_1(X_1), t_2(X_1, X_2), \ldots, t_s(X_1, \ldots, X_s)$, then T generates a zero-dimensional ideal which is equal to $\mathrm{sat}(T)$. If, in addition, the ideal $\mathrm{sat}(T)$ is prime (and, thus maximal in this case), then T is an encoding of the field extension $\mathbb{L} := \mathbb{K}[X]/\langle T \rangle$. Let $H \subset \mathbb{K}[X]$. The pair $[T, H]$ is a *regular system* if each polynomial in H is regular modulo $\mathrm{sat}(T)$; the zero set of $[T, H]$, denoted by $Z(T, H)$, consists of all points of $\mathbb{K}^s$ satisfying $t = 0$ for all $t \in T$, $h \neq 0$ for all $h \in H \cup \{h_T\}$. A regular chain T, or a regular system $[T, H]$, is *square-free* if for all $t \in T$, the polynomial $\mathrm{der}(t)$ is regular w.r.t. $\mathrm{sat}(T)$, where $\mathrm{der}(t) = \frac{\partial t}{\partial v}$ and $v = \mathrm{mvar}(t)$.

The zero set S of an arbitrary system of polynomial equations and inequations is called a *constructible set* and can be decomposed as the union of the zero sets of finitely many square-free regular systems $[T_1, H_1], \ldots, [T_e, H_e]$. When this holds we have $S = Z(T_1, H_1) \cup \cdots \cup Z(T_e, H_e)$ and we say that $[T_1, H_1], \ldots, [T_e, H_e]$ is a *triangular decomposition* of S.

We specify below a core routine thanks to which triangular decompositions can be computed. For more details about the theory of regular chains and its algorithmic aspects, we refer to [30].

Notation 1. *The function* $\mathsf{Squarefree\_RC}(p, T, H)$ *computes a set of triples* $((b_{i,1}, \ldots, b_{i,\ell_i}), T_i, H_i)$ *with* $1 \leq i \leq e$, *such that* $[T_1, H_1], \ldots, [T_e, H_e]$ *are regular systems forming a triangular decomposition of* $Z(T, H)$, *and for all* $1 \leq i \leq e$:

1. $b_{i,1}, \ldots, b_{i,\ell_i}$ *are polynomials with the same main variable* $v = \mathrm{mvar}(p)$ *such that we have* $p \equiv \prod_{j=1}^{\ell_i} b_{i,j}^j \mod \mathrm{sat}(T_i)$,
2. *all discriminants* $\mathrm{discr}(b_{i,j}, v)$ *and all resultants* $\mathrm{res}(b_{i,j}, b_{i,k}, v)$ *are regular modulo* $\mathrm{sat}(T_i)$, *thus* $\prod_{j=1}^{\ell_i} b_{i,j}^j$ *is a square-free factorization of* p *modulo* $\mathrm{sat}(T_i)$.

3.2 Regular Chain Representation of a Splitting Field

Let $p(x) \in K[x]$ be a monic univariate polynomial. The *splitting field* of $p(x)$ over $\mathbb{K}$ is the smallest field extension of $\mathbb{K}$ over which $p(x)$ splits into linear factors,

$$p(x) = \prod_{i=1}^{\ell} (x - r_i)^{m_i}. \tag{1}$$

The set $\{r_1, \ldots, r_\ell\}$ generates $\mathbb{L}$ over $\mathbb{K}$. That is, $\mathbb{L} = \mathbb{K}(r_1, \ldots, r_\ell)$.

Assume that $p(x)$ is an irreducible, monic polynomial in $\mathbb{K}[x]$ of degree $n \geq 2$. To construct the splitting field $\mathbb{L}$ of $p(x)$ and compute the factorization of $p(x)$ into linear factors over $\mathbb{L}$, we proceed as follows.

1. Initialize $i := 1$, $y_i := x$, $\mathbb{L} := \mathbb{K}$, $T := \{\}$, $\mathcal{P} := \{\}$ and $\mathcal{F} := \{p\}$; the set $\mathcal{F}$ is assumed to maintain a list of univariate polynomials in y_i, irreducible over the current value of $\mathbb{L}$ and, of degree at least two,
2. While $\mathcal{F}$ is not empty do
   (S1) pick a polynomial $f(y_i) \in \mathcal{F}$ over $\mathbb{L}$,
   (S2) let α_i be a root of $f(y_i)$ (in the algebraic closure of $\mathbb{K}$),
   (S3) replace $\mathbb{L}$ by $\mathbb{L}(\alpha_i)$, that is, by adjoining α_i to $\mathbb{L}$,
   (S4) replace T by $T \cup \{t_i(y_1, \ldots, y_i)\}$, where the multivariate $t_i(y_1, \ldots, y_i)$ is obtained from $f(y_i)$ after replacing the algebraic numbers $\alpha_1, \ldots, \alpha_{i-1}$ with the variables $y_1, \ldots, y_{i-1}$,
   (S5) replace $\mathcal{P}$ by $\mathcal{P} \cup \{x - y_i\}$,
   (S6) factor $f(y_i)$ into irreducible factors over $\mathbb{L}$, then add the obtained factors of degree 1 (resp. greater than 1) to $\mathcal{P}$ (resp. $\mathcal{F}$); when adding a factor to $\mathcal{P}$, replace $\alpha_1, \ldots, \alpha_{i-1}$ with $y_1, \ldots, y_{i-1}$; when adding a factor to $\mathcal{F}$, replace y_i with y_{i+1}.
   (S7) if $\mathcal{F}$ is not empty then $i := i + 1$.
3. Set $s := i$ and return $(s, T, \mathcal{P})$.

At the end of this procedure, the set T is a regular chain in the polynomial ring $\mathbb{K}[y_1, \ldots, y_s]$ generating a maximal ideal such that $\mathbb{K}[y_1, \ldots, y_s]/\langle T \rangle$ is isomorphic to the splitting field $\mathbb{K}(p)$ of $p(x)$. This procedure can be derived from Landau's paper [31]; note that the factorization at Step (S6) can be performed, for instance, by the algorithm of Trager [32]. Example: with $p(x) = x^3 - 2$, one can find $T = \{y_1^3 - 2, y_2^2 + y_1 y_2 + y_1^2\}$ and $\mathcal{P} = \{x - y_1, x - y_2, x + y_2 + y_1\}$.

3.3 The Frobenius Canonical Form

Throughout the sequel of this section, we denote by A a square matrix of dimension n with entries in a field $\mathbb{K}$.

Let $p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ be a monic polynomial in $\mathbb{K}[x]$. The *Frobenius companion matrix*[1] of $p(x)$ is a square $n \times n$ matrix of the form

$$C(p(x)) = \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & \cdots & 0 & -a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_{n-1} \end{bmatrix}. \tag{2}$$

A matrix $F \in \mathbb{K}^{n\times n}$ is said to be in *Frobenius (rational) canonical form* if it is a block diagonal matrix where the blocks are companion matrices of monic polynomials $\psi_i(x) \in \mathbb{K}[x]$

$$F = \bigoplus_{i=1}^{m} C(\psi_i(x)) \tag{3}$$

such that $\psi_{i-1} \mid \psi_i$ for $i = 1, \ldots, m-1$. The polynomials ψ_i are the *invariant factors* of F. We recall a few properties below, see [5,33,34] for details:

1. Every companion matrix is in Frobenius canonical form.
2. For all $i = 1, \ldots, m$, the companion matrix $C(\psi_i)$ is non-derogatory[2].
3. There exists a nonsingular matrix $Q \in \mathbb{K}^{n\times n}$ such that $F := Q^{-1}AQ$ is in Frobenius canonical form. The matrix F is called the *Frobenius canonical form* of A and the matrices A and F are said to be *similar*. We note that A and F have the same invariant factors.
4. The polynomial ψ_1 is the *minimal polynomial* of F and the product $\prod \psi_i$ is the *characteristic polynomial* of F.

3.4 The Jordan Canonical Form

An element $\lambda \in \overline{\mathbb{K}}$ is an *eigenvalue* of A if it satisfies $\det(A - \lambda I_n) = 0$ where I_n is the identity matrix of dimension n. The *algebraic multiplicity* of an eigenvalue λ is its multiplicity as a root of the characteristic polynomial of A, and its *geometric multiplicity* is the dimension of the null space of $A - \lambda I_n$.

Let $F = \mathrm{diag}(C(\psi_1), C(\psi_2), \ldots, C(\psi_m))$ be the Frobenius form of A where $C(\psi_i)$ is the companion matrix of the ith invariant factor ψ_i of A. We note that the geometric multiplicity of an eigenvalue λ of A is the number of invariant factors that λ is a solution for. Thus, the Frobenius form of A tells us both the algebraic and geometric multiplicities of all eigenvalues of A.

[1] There are many other companion matrices, but in this paper a "companion matrix" is a Frobenius companion matrix.

[2] The characteristic polynomial and the minimal polynomial coincide up to a factor of ± 1.

A matrix is called a *Jordan block* of dimension n if it is zero everywhere except for ones along its superdiagonal, and a single value λ along its main diagonal. A Jordan block has one eigenvalue λ with geometric multiplicity 1 and algebraic multiplicity n. We use the notation $\mathrm{JBM}_n(\lambda)$ to denote a Jordan block of dimension n with eigenvalue λ.

Let F be a matrix in Frobenius form as in Eq. (3). The *Jordan canonical form* of F is given by

$$J = \bigoplus_{i=1}^{m} \mathrm{JCF}(C(\psi_i(x))) \tag{4}$$

where $\mathrm{JCF}(C(\psi(x)))$ is the Jordan form of a companion matrix of $\psi(x)$, see Chapter VI, Sect. 6 of [33] for a proof.

4 JCF over a Splitting Field

4.1 Jordan Form of a Companion Matrix

Let $\psi(x) \in \mathbb{K}[x]$ be a univariate monic polynomial of degree n. Let $\mathbb{L}$ be the splitting field of $\psi(x)$ over $\mathbb{K}$. Let $C = C(\psi(x))$ be the companion matrix of $\psi(x)$. Assume that the complete factorization into linear factors of $\psi(x)$ writes

$$\psi(x) = \prod_{i=1}^{\ell} (x - r_i)^{m_i} \tag{5}$$

where $r_i \in \mathbb{L}$ for $i = 1 \ldots \ell$ and $r_i \neq r_j$ for $i \neq j$. Then, the Jordan form of C is given by

$$J = \bigoplus_{i=1}^{\ell} \mathrm{JBM}_{m_i}(r_i) \tag{6}$$

where the entries of J are in $\mathbb{L}$. Thus, once the splitting field of $\psi(x)$ is computed, the Jordan canonical form of the companion matrix of $\psi(x)$ can be constructed.

Using the algorithm described in Sect. 3.2, the roots $r_1, \ldots, r_\ell$ of $\psi(x)$ are represented by the residue classes of multivariate polynomials $r_1(y_1, \ldots, y_s), \ldots, r_\ell(y_1, \ldots, y_s)$ modulo $\langle T \rangle$, since the regular chain $T = t_1(y_1), \ldots, t_s(y_1, \ldots, y_s)$ encodes the splitting field $\mathbb{K}(\psi)$ of $\psi(x)$ in the sense that this field is isomorphic to $\mathbb{K}[y_1, \ldots, y_i]/\langle T \rangle$. Therefore, the Jordan form of C is given by

$$\bigoplus_{i=1}^{\ell} \mathrm{JBM}_{m_i}(r_i(y_1, \ldots, y_s)) \tag{7}$$

together with the regular chain T.

4.2 Frobenius Form to Jordan Form

Let $F \in \mathbb{K}^{n \times n}$ be in Frobenius form, with $F = \mathrm{diag}(C(\psi_1), C(\psi_2), \ldots, C(\psi_m))$, where the polynomials ψ_i are the invariant factors of F. By Eq. (4), the Jordan form of F is given by $J = \bigoplus_{i=1}^{m} \mathrm{JCF}(C(\psi_i))$ and a regular chain T defining the splitting field of ψ_1. This is, indeed, sufficient to compute all the entries of the JCF of F, since every subsequent polynomial ψ_i divides ψ_1.

4.3 Example

Let $\psi(x) = (x^3 + x^2 + x - 1)(x^2 + x + 1)^2$, where the coefficients are in $\mathbb{Q}$. Let C be the companion matrix of ψ. The JCF of C over the splitting field $\mathbb{L}$ of ψ over $\mathbb{Q}$ is

$$\begin{bmatrix} y_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & y_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 - y_1 - y_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & y_2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & y_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 - y_2 & 1 \\ 0 & 0 & 0 & 0 & 0 & & -1 - y_2 \end{bmatrix}$$

where (y_1, y_2, y_3) are any point in the zero set $V(T)$ where T is

$$T = \{y_1^2 + (1 + y_3)y_1 + y_3^2 + y_3 + 1, y_2^2 + y_2 + 1, y_3^3 + y_3^2 + y_3 - 1\}.$$

5 JCF of a Matrix with Parameters

In this section we show how to compute a complete case discussion for the JCF of a matrix F in Frobenius form where the entries are polynomials in $\mathbb{K}[\alpha_1, \ldots, \alpha_s]$. Note that, as Arnol'd points out in [16], a parametric Frobenius form is continuous in its parameters, though its Jordan form may not be. Throughout this section, $T \subset \mathbb{K}[\alpha]$ will be a regular chain and $H \subset \mathbb{K}[\alpha]$ a set of polynomial inequations such that $[T, H]$ forms a regular system.

5.1 Square-Free Factorization of a Parametric Polynomial

Let $\alpha_1 < \cdots < \alpha_s$ be $s \geq 1$ ordered variables. Let $\mathbb{K}[\alpha] = \mathbb{K}[\alpha_1, \ldots, \alpha_s]$ be the ring of polynomials in the variables $\alpha = \alpha_1, \ldots, \alpha_s$. Let x be a variable. Let $\mathbb{K}[x]$ (resp. $\mathbb{K}[\alpha][x]$) be the ring of polynomials in x with coefficients in $\mathbb{K}$ (resp. $\mathbb{K}[\alpha]$). A polynomial $p(x; \alpha) \in \mathbb{K}[\alpha][x]$ is called a *univariate, parametric polynomial* in x and takes the form

$$p(x; \alpha) = a_n(\alpha)x^n + \cdots + a_1(\alpha)x + a_0(\alpha) \tag{8}$$

where the coefficients $a_i(\alpha)$ are polynomials in $\mathbb{K}[\alpha]$.

Let $p(x; \alpha) = \prod_{i=1}^{\ell} b_i(x; \alpha)^i$ be a square-free factorization of $p(x; \alpha)$, regarded as a univariate polynomial in $\mathbb{K}[\alpha][x]$. Then, the following properties must hold:

1. each polynomial $b_i(x; \alpha)$ is square-free as a polynomial in $\mathbb{K}[\alpha][x]$, and
2. the GCD of $b_i(x; \alpha)$ and $b_j(x; \alpha)$, as polynomials in $\mathbb{K}[\alpha][x]$, has degree zero in x, for all $1 \leq i < j \leq \ell$.

We note that each of the square-free factors $b_1, \ldots, b_\ell$ of $p(x; \alpha)$ is uniquely defined up to a multiplicative element of $\mathbb{K}[\alpha]$.

Definition 1. *We say that the sequence of polynomials* $b_1, \ldots, b_\ell$ specializes well *at a point* $\alpha^* = (\alpha_1^*, \ldots, \alpha_s^*) \in \overline{\mathbb{K}}^s$ *whenever*

1. *the degree in* x *of the specialized polynomial* $b_i(x; \alpha^*)$ *is the same as the degree in* x *of* b_i *as a polynomial in* $\mathbb{K}[\alpha][x]$*, for all* $1 \leq i \leq \ell$*;*
2. *each specialized polynomial* $b_i(x; \alpha^*)$ *is square-free, as a polynomial in* $\mathbb{K}[x]$*, for all* $1 \leq i \leq \ell$*; and*
3. *the GCD of* $b_i(x; \alpha^*)$ *and* $b_j(x; \alpha^*)$*, as polynomials in* $\mathbb{K}[x]$*, has degree zero in* x*, for all* $1 \leq i < j \leq \ell$*.*

From the theory of *border polynomials* [27–29] the following result holds.

Proposition 1. *The set of points* $\alpha \in \overline{\mathbb{K}}^s$ *at which the sequence of polynomials* $b_1, \ldots, b_\ell$ *specializes well is the complement of the algebraic set given by*

$$\left\{ \bigcup_{i=1}^{i=e} V(\Delta_i) \right\} \cup \left\{ \bigcup_{1 \leq i < j \leq e} V(R_{i,j}) \right\}, \tag{9}$$

where $\Delta_i := \operatorname{discr}(b_i(x; \alpha), x)$ *denotes the discriminant of* $b_i(x; \alpha)$ *w.r.t.* x *and* $R_{i,j} := \operatorname{res}(b_i(x; \alpha), b_j(x; \alpha), x)$ *denotes the resultant of* $b_i(x; \alpha)$ *and* $b_j(x; \alpha)$ *w.r.t.* x*.*

Definition 2. *We call the* proviso *of the sequence of polynomials* $b_1, \ldots, b_\ell$ *the algebraic set (actually hypersurface) given by Eq.* (9) *and denote it by* $\mathsf{Proviso}(b_1, \ldots, b_\ell)$*. We call the* square-free factorization with proviso *of* $p(x; \alpha)$ *the pair* $(\prod_{i=1}^{\ell} b_i(x; \alpha)^i, \mathsf{Proviso}(b_1, \ldots, b_\ell))$*.*

We note that the zero set of the border polynomial of $p(x; \alpha)$ (in the sense [27,29]) is usually defined whenever $p(x; \alpha)$ is square-free w.r.t. x, in which case it coincides with $\mathsf{Proviso}(b_1, \ldots, b_\ell)$.

We are now interested in obtaining a complete case discussion for the square-free factorization of $p(x; \alpha)$, that is, including the cases where $\alpha^* \in \mathsf{Proviso}(p(x; \alpha), x)$ holds. This can be achieved by using the function $\mathsf{Squarefree\_RC}(p, T, H)$ specified in Sect. 3.1.

5.2 JCF of a Companion Matrix with Parameters

From now on, we assume that the field $\overline{\mathbb{K}}$ is $\mathbb{C}$. Let $C \in \mathbb{K}[\alpha]^{n \times n}$ be a companion matrix with characteristic polynomial $\psi(x; \alpha) \in \mathbb{K}[\alpha][x]$. Let $\prod_{i=1}^{\ell} b_i(x; \alpha)^i$ be a square-free factorization of $\psi(x; \alpha)$. We observe that in the complement of $\mathsf{Proviso}(b_1, \ldots, b_\ell)$, the roots (in x) of $b_1, \ldots, b_\ell$, as functions of α, define continuous, disjoint graphs. Let us denote those functions by $\lambda_{i,1}, \ldots, \lambda_{i,n_i}$ corresponding to the polynomial b_i, for $1 \leq i \leq \ell$. Therefore, one can construct the JCF of C uniformly over the complement of $\mathsf{Proviso}(b_1, \ldots, b_\ell)$ as follows

$$\bigoplus_{i=1}^{\ell} \bigoplus_{j=1}^{n_i} \mathrm{JBM}_i(\lambda_{i,j}). \tag{10}$$

More generally, for a regular system $[T, H]$ let $((b_{i,1}, \ldots, b_{i,\ell_i}), T_i, H_i)$, with $1 \leq i \leq e$, be the output of $\mathsf{Squarefree\_RC}(\psi(x;\alpha), T, H)$. Then, for every $1 \leq i \leq e$, one can construct the JCF of C uniformly over $Z(T_i, H_i)$ as the regular systems $[T_1, H_1], \ldots, [T_e, H_e]$ form a triangular decomposition of $Z(T, H)$.

5.3 Frobenius Form to JCF with Parameters

Let $F \in \mathbb{K}[\alpha]^{n\times n}$ be a matrix in Frobenius form with invariant factors $\psi_i(x;\alpha) \in \mathbb{K}[\alpha][x]$ for $1 \leq i \leq m$. Let $\prod_{i=1}^{\ell} b_i(x;\alpha)^i$ be a square-free factorization of the minimal polynomial, $\psi_1(x;\alpha)$. The JCF over the complement of $\mathsf{Proviso}(b_1, \ldots, b_\ell)$ is defined continuously for each companion matrix $C(\psi_i(x;\alpha))$, $1 \leq i \leq m$. This is a consequence of the property that each subsequent $\psi_i(x;\alpha)$ divides $\psi_1(x;\alpha)$.

The construction of the JCF of $C(\psi_1(x;\alpha))$ defines a decomposition of the complement of $\mathsf{Proviso}(b_1, \ldots, b_\ell)$ into the zero sets of finitely many square-free regular systems $[T_1, H_1], \ldots, [T_e, H_e]$. Over each regular system, the JCF of each companion matrix $C(\psi_i(x;\alpha))$ for $1 \leq i \leq m$ is defined continuously.

6 Experimentation

We are actively developing a package called `ParametricMatrixTools` in Maple that implements algorithms for computations on matrices with parameters. The source for this package, including numerous examples, is available at github.com/StevenThornton/ParametricMatrixTools and is compatible the version of the `RegularChains` library included in Maple 2016 and later. The `ComprehensiveJordanForm` method implements the algorithm discussed in Sect. 5. Further details can be found at regularchains.org.

For each of the examples that follow, we have first computed a full case discussion for the Frobenius form using the `ComprehensiveFrobeniusForm` routine in our package. The details of the Frobenius form implementation have been omitted and we are actively working to improve our current implementation.

6.1 Kac-Murdock-Szegö matrices

The inverse matrix $K_n^{-1}(\rho)$ from [35] is

$$\frac{1}{1-\rho^2}\begin{bmatrix} 1 & -\rho & 0 & \cdots & 0 & 0 & 0 \\ -\rho & 1+\rho^2 & -\rho & \cdots & 0 & 0 & 0 \\ 0 & -\rho & 1+\rho^2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1+\rho^2 & -\rho & 0 \\ 0 & 0 & 0 & \cdots & -\rho & 1+\rho^2 & -\rho \\ 0 & 0 & 0 & \cdots & 0 & -\rho & 1 \end{bmatrix}.$$

The cost to compute a full case discussion of the JCF of $(1-\rho^2)K_n^{-1}(\rho)$ grows exponentially with n. See Fig. 2.

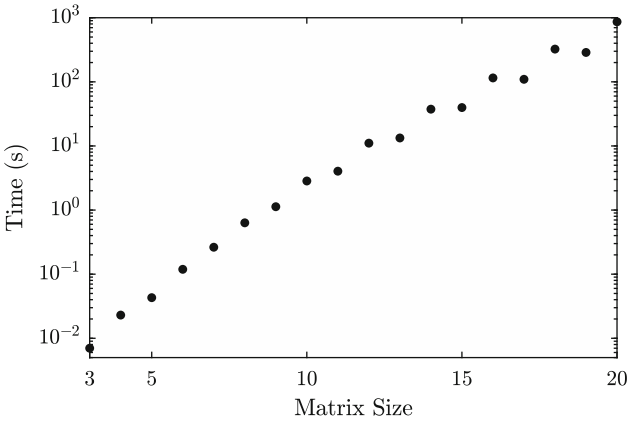


Fig. 2. Time to compute the JCF of each Frobenius form in the full case discussion of the Frobenius form of the matrix in Sect. 6.1. For all n, the Frobenius form splits into two cases: $\rho = 0$ and $\rho \neq 0$. The JCF is computed over each of these branches. Note the exponential growth. Timing was done on a 2016, 3.3 GHz quad-core Intel Core i7 iMac with 16 GB of RAM using Maple 2016.2.

6.2 The Belousov-Zhabotinskii Reaction

The report [36] contains a very readable account of the famous B-Z reaction and its history. This is a chemical oscillator. In non-dimensional form with $\varepsilon = \delta = 1$ we have

$$\begin{aligned} \dot{x} &= qy - xy + x(1-x) \\ \dot{y} &= -qy - xy + fz \\ \dot{z} &= x - z \end{aligned}$$

The equilibria include $x = z$ being a positive root of the quadratic

$$x(x - 1 + f) + q(x - 1 - f) = 0. \tag{11}$$

The Jacobian at the equilibrium is

$$A = \begin{bmatrix} 1-x-y & q-x & 0 \\ -y & -(q+x) & f \\ 1 & 0 & -1 \end{bmatrix} \tag{12}$$

and the Jordan form of A splits into many cases. One non-trivial example is

$$J = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 1 \\ 0 & 0 & \beta \end{bmatrix} \tag{13}$$

where

$$\alpha = \frac{1}{9994}(-81q^5 + 804q^4 - 3882q^3 + 12209q^2 - 6288q - 59636)$$
$$\beta = \frac{1}{2}(-\alpha + 3q - 10)$$

under the following constraints on the indeterminates of A:

$$x = z = -2y$$
$$f = -1$$
$$(q^5 - 13q^4 + 86q^3 - 359q^2 + 911q - 742)z - 4q^2 - 8 = 0$$
$$q^6 - 15q^5 + 112q^4 - 531q^3 + 1633q^2 - 2564q + 1492 = 0.$$

There are real values of q satisfying this equation, and hence this case is real.

6.3 Nuclear Magnetic Resonance

In [6], Sect. 2.2, we find a concise description of an application of the matrix exponential to solve the so-called Solomon equations

$$\dot{M} = -RM, \quad M(0) = I \quad \text{by} \quad M(t) = e^{-Rt}. \tag{14}$$

Here R is a symmetric, diagonally dominant matrix called the relaxation matrix, and M is the matrix of intensities. Suppose R is in fact tridiagonal, with ones on the sub- and super-diagonals, and diagonal parameters $|r_i| > 1$. Using MAPLE's built-in `MatrixExponential` gets answers (e.g. when the dimension n is 3) but we are not convinced that the generic answer returned is correct, always. So we try computing the JCF. Doing so, we find that indeed there are special cases that the generic code missed. For example, when R is of dimension 3, the JCF of R is

$$\begin{bmatrix} (r_1 + r_2 + r_3)/3 & 1 & \\ 0 & (r_1 + r_2 + r_3)/3 & 1 \\ 0 & 0 & (r_1 + r_2 + r_3)/3 \end{bmatrix} \tag{15}$$

when

$$r_1^2 + r_2^2 + r_3^2 - r_1r_2 - r_1r_3 - r_2r_3 + 6 = 0 \tag{16}$$
$$((r_1 - r_3)^2 - 1)((r_1 - r_3)^2 + 8) = 0. \tag{17}$$

When $\mathrm{discr}(\texttt{CharPoly}(A)) \neq 0$ the JCF is simply $\mathrm{diag}(\lambda_1, \lambda_2, \lambda_3)$ for the distinct eigenvalues $\lambda_1, \lambda_2, \lambda_3$. And for the remaining parameter values, the JCF consists of a Jordan block of dimension 2 with eigenvalue λ_1, and a Jordan block of dimension 1 with eigenvalue λ_2 for $\lambda_1 \neq \lambda_2$. The only case corresponding to real values of r_1, r_2, r_3 is the trivial diagonal case. In the cases where the JCF is not a diagonal matrix, the result computed by the `MatrixExponential` function in MAPLE contains discontinuities.

6.4 Bifurcation Studies

The mathematical methods used in bifurcation studies are highly sophisticated, both symbolically and numerically. Tools used include normal forms and the action of symmetry groups. Consider the matrix

$$J = \begin{bmatrix} 0 & 2\rho & 0 \\ a & 2\beta & 2v \\ b & -2v & 2\beta \end{bmatrix} \tag{18}$$

which is the Jacobian matrix of a dynamical system at equilibrium. The analysis of this system in [12] is quite complete, yet the evolution of trajectories near the equilibria, governed by

$$\xi' = J\xi, \quad \xi(0) = I \tag{19}$$

or $\xi = \exp(tJ)$, is of interest. When the JCF of J is nontrivial, one can anticipate phenomena such as greater sensitivity to modelling error, for instance. Our implementation is able to find a complete case discussion of the JCF, starting from the complete case discussion of the Frobenius form, in approximately 2 seconds. We find cases corresponding to each of the 5 possible Jordan structures for a 3×3 matrix with a total of 46 cases. Of the 46 cases, 14 are defined by polynomials of total degree greater than 4. The worst case contains a polynomial of degree 12 in the parameters with 19 terms.

One non-trivial case we were able to automatically identify is where the JCF of J is given by

$$\begin{bmatrix} 2\beta & 0 & 0 \\ 0 & \beta & 1 \\ 0 & 0 & \beta \end{bmatrix} \tag{20}$$

when $2\rho a + \beta^2 = 0$, $v = 0$, and a, ρ and β are non-zero.

References

1. Hogben, L.: Handbook of Linear Algebra. CRC Press, Boca Raton (2016)
2. O'Meara, K., Clark, J., Vinsonhaler, C.I.: Advanced Topics in Linear Algebra: Weaving Matrix Problems Through the Weyr Form. Oxford University Press, New York (2011)
3. Horn, R.A., Johnson, C.R.: Matrix Analysis. CUP (2012)
4. Giesbrecht, M.: Nearly optimal algorithms for canonical matrix forms. SIAM J. Comput. **24**(5), 948–969 (1995)
5. Kaltofen, E., Krishnamoorthy, M., Saunders, B.D.: Fast parallel algorithms for similarity of matrices. In: Proceedings of the Fifth ACM Symposium on Symbolic and Algebraic Computation, pp. 65–70. ACM (1986)
6. Higham, N.J.: Functions of Matrices: Theory and Computation. SIAM, Philadelphia (2008)
7. Moler, C., Van Loan, C.: Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev. **45**(1), 3–49 (2003)
8. Frank, J., Huang, W., Leimkuhler, B.: Geometric integrators for classical spin systems. J. Comput. Phys. **133**(1), 160–172 (1997)

9. Hairer, E., Lubich, C., Wanner, G.: Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations, vol. 31. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-30666-8
10. Kunkel, P., Mehrmann, V.: Differential-Algebraic Equations Analysis and Numerical Solution. EMS Publishing House, Zürich (2006)
11. Corless, R.M., Jeffrey, D.J.: Well...it isn't quite that simple. ACM SIGSAM Bull. **26**(3), 2–6 (1992)
12. Van Gils, S., Krupa, M., Langford, W.F.: Hopf bifurcation with non-semisimple 1:1 resonance. Nonlinearity **3**(3), 825 (1990)
13. Beelen, T., Van Dooren, P.: Computational Aspects of the Jordan Canonical Form, pp. 57–72. Oxford Science Publication, Oxford University Press, New York (1990)
14. Gil, I.: Computation of the Jordan canonical form of a square matrix (using the Axiom programming language), pp. 138–145. ACM (1992)
15. Ozello, P.: Calcul exact des formes de Jordan et de Frobenius d'une matrice. Ph.D. thesis, Université Joseph-Fourier-Grenoble I (1987)
16. Arnol'd, V.I.: On matrices depending on parameters. Russian Math. Surv. **26**(2), 29–43 (1971)
17. Chen, G., Della Dora, J.: Rational normal form for dynamical systems by Carleman linearization. In: Proceedings of ISSAC 1999, pp. 165–172. ACM (1999)
18. Storjohann, A.: An $\mathcal{O}(n^3)$ algorithm for the Frobenius normal form. In: Proceedings of ISSAC 1998, pp. 101–105. ACM (1998)
19. Chen, G.: Computing the normal forms of matrices depending on parameters. In: Proceedings of ISSAC 1989, pp. 242–249. ACM (1989)
20. Storjohann, A.: Algorithms for matrix canonical forms. Ph.D. thesis, Swiss Federal Institute of Technology Zurich (2013)
21. Ballarin, C., Kauers, M.: Solving parametric linear systems. ACM SIGSAM Bull. **38**(2), 33–46 (2004)
22. Broadbery, P.A., Gómez-Díaz, T., Watt, S.M.: On the implementation of dynamic evaluation. In: Proceedings of ISSAC 1995, pp. 77–84. ACM (1995)
23. Diaz-Toca, G.M., Gonzalez-Vega, L., Lombardi, H.: Generalizing Cramer's rule. SIAM J. Matrix Anal. Appl. **27**(3), 621–637 (2005)
24. Kapur, D.: An approach for solving systems of parametric polynomial equations. In: Principles and Practices of Constraint Programming, pp. 217–244 (1995)
25. Sit, W.Y.: An algorithm for solving parametric linear systems. J. Symb. Comput. **13**(4), 353–394 (1992)
26. Corless, R.M., Thornton, S.E.: A package for parametric matrix computations. In: Mathematical Software - ICMS 2014–4th International Congress, Seoul, South Korea, 5–9 August 2014, Proceedings, pp. 442–449 (2014)
27. Yang, L., Hou, X., Xia, B.: A complete algorithm for automated discovering of a class of inequality-type theorems. Sci. China Ser. F Inf. Sci. **44**(1), 33–49 (2001)
28. Lazard, D., Rouillier, F.: Solving parametric polynomial systems. J. Symb. Comput. **42**(6), 636–667 (2007)
29. Moreno Maza, M., Xia, B., Xiao, R.: On solving parametric polynomial systems. Math. Comput. Sci. **6**(4), 457–473 (2012)
30. Chen, C., Moreno Maza, M.: Algorithms for computing triangular decomposition of polynomial systems. J. Symb. Comput. **47**(6), 610–642 (2012)
31. Landau, S.: Factoring polynomials over algebraic number fields. SIAM J. Comput. **14**(1), 184–195 (1985)

32. Trager, B.M.: Algebraic factoring and rational function integration. In: Jenks, R.D. (ed.) SYMSAC 1976, Proceedings of the Third ACM Symposium on Symbolic and Algebraic Manipulation, Yorktown Heights, New York, USA, 10–12 August 1976. ACM, pp. 219–226 (1976)
33. Gantmacher, F.R.: The Theory of Matrices, vol. 1. Chelsea Publishing Company, New York (1960)
34. Gohberg, I., Lancaster, P., Rodman, L.: Matrix Polynomials. SIAM, Philadelphia (2009)
35. Trench, W.F.: Properties of some generalizations of KAC-Murdock-Szegö matrices. In: Structured Matrices in Mathematics, Computer Science II Control, Signal and Image Processing (AMS Contemporary Mathematics Series), vol. 281 (2001)
36. Gray, C.R.: An analysis of the Belousov-Zhabotinskii reaction. Rose-Hulman Undergraduate Math. J. **3**(1) (2002)

Knowledge-Based Interoperability for Mathematical Software Systems

Michael Kohlhase[1], Luca De Feo[5], Dennis Müller[1(✉)], Markus Pfeiffer[3], Florian Rabe[2], Nicolas M. Thiéry[4], Victor Vasilyev[3], and Tom Wiesing[1]

[1] FAU Erlangen-Nürnberg, Erlangen, Germany
d.mueller@kwarc.info
[2] Jacobs University Bremen, Bremen, Germany
[3] University of St Andrews, St Andrews, UK
[4] Université Paris-Sud, Orsay, France
[5] Université Versailles St Quentin, Versailles, France

Abstract. There is a large ecosystem of mathematical software systems. Individually, these are optimized for particular domains and functionalities, and together they cover many needs of practical and theoretical mathematics. However, each system specializes on one particular area, and it remains very difficult to solve problems that need to involve multiple systems. Some integrations exist, but the are ad-hoc and have scalability and maintainability issues. In particular, there is not yet an interoperability layer that combines the various systems into a virtual research environment (VRE) for mathematics.

The OpenDreamKit project aims at building a toolkit for such VREs. It suggests using a central system-agnostic formalization of mathematics (Math-in-the-Middle, MitM) as the needed interoperability layer. In this paper, we report on a case study that instantiates the MitM paradigm the systems GAP, SageMath, and Singular to perform computation in group and ring theory.

Our work involves massive practical efforts, including a novel formalization of computational group theory, improvements to the involved software systems, and a novel mediating system that sits at the center of a star-shaped integration layout between mathematical software systems.

1 Introduction

There is a large and vibrant ecosystem of open-source software systems for mathematics. These range from calculators, which perform simple computations, via mathematical databases, which curate collections of a mathematical objects, to powerful modeling tools and computer algebra systems (CAS).

Most of these systems are very specific – they focus on one or very few aspects of mathematics. For example, among databases, the "Online Encyclopedia of Integer Sequences" (OEIS) focuses on sequences over $\mathbb{Z}$ and their properties, and the "L-Functions and Modular Forms Database" (LMFDB) [Cre16, LMFDB] on objects in number theory pertaining to Langland's program. Among

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 195–210, 2017.
https://doi.org/10.1007/978-3-319-72453-9_14

CAS, GAP [GAP] excels at discrete algebra with a focus on group theory, Singular [SNG] focuses on polynomial computations with special emphasis on commutative and non-commutative algebra, algebraic geometry, and singularity theory, and SageMath [Sage] aims to be a general purpose software for computational pure mathematics by loosely integrating many systems including the aforementioned ones.

For a mathematician, however, (a user, which we call Jane) the systems themselves are not relevant. Instead, she only cares about being able to solve problems. Because it is typically not possible to solve a mathematical problem using only a single program, Jane has to work with multiple systems and combine the results to reach a solution. Currently there is very little tool support for this practice, so Jane has to isolate sub-problems that the respective systems are amenable to, formulate them in the respective input language, collect intermediate results and reformulate them for the next system – a tedious and error-prone process at best, a significant impediment to scientific progress at worst. Solutions for some situations certainly exist, which can help get Jane unstuck, but these are ad-hoc and only for specific often-used system combinations. Moreover, each of these ad hoc solutions requires a lot of maintenance and scales badly to multi-system integration.

One goal of the OpenDreamKit project is tackling these problems systematically by building virtual research environments (VRE) on top of the existing systems. To build a VRE from individual systems, we need a joint user interface – the OpenDreamKit project adopts Jupyter [Jup] and active documents [Koh+11] – and an interoperability layer that allows passing problems and results between the disparate systems. For the latter, it proposes the Math-in-the-Middle (MitM [Deh+16]) paradigm, an interoperability framework based on a central, system-independent ontology of mathematical knowledge. In this paper we instantiate the MitM paradigm in a concrete case study using a distributed computation involving GAP, SageMath, and Singular.

We will use the following running example from computational group theory: Jane wants to experiment with invariant theory of finite groups. She works in the polynomial ring $R = \mathbb{Z}[X_1, \ldots, X_n]$, and wants to construct an ideal I in this ring that is fixed by a group $G \leq S_n$ acting on the variables, linking properties of the group to properties of I and the quotient of R by I.

To construct an ideal that is invariant under the group action, it is natural to pick some polynomial p from R and consider the ideal I of R that is generated by all elements of the orbit $O = Orbit(G, R, p) \subseteq R$. For effective further computation with I, she needs a Göbner base of I.

Jane is a SageMath user and wants to receive the result in SageMath, but she wants to use GAP's orbit algorithm and Singular's Gröbner base algorithm, which she knows to be very efficient. For the sake of example, we will work with $n = 4$, $G = D_4$ (the dihedral group[1]), and $p = 3 \cdot X_1 + 2 \cdot X_2$, but our results apply to arbitrary values.

[1] Incidentally, this group is called D_4 in SageMath but D_8 in GAP due to differing conventions in different mathematical communities – a small example of the obstacles to system interoperability that MitM tackles.

In Sect. 2, we recap the MitM paradigm. MitM solutions consist of three parts: a central ontology, specifications of the abstract languages of the involved systems (which we call *system dialects*), and the distributed computation infrastructure that connects the systems via the ontology as an intermediate representation. The rest of the paper develops these three parts for our case study: In Sect. 3, we contribute a fragment to the MitM ontology that formalizes computational group theory. In Sect. 4, we specify the abstract languages of GAP, SageMath, and Singular and their relation to the ontology. Finally in Sect. 5, we present the resulting virtual research environment built on these systems in action. Section 6 concludes the paper and compares MitM-based interoperability with other approaches.

2 Math-in-the-Middle Interoperability

Figure 1 shows the basic MitM design. We want to make the systems A to H with system dialects a to h interoperable. A P2P translation regime ($n(n-1)$ translations between n systems) is already intractable for the systems in the OpenDreamKit project (more than a dozen). Alternatively, an "industry standard" regime, where one system dialect is declared as the standard is infeasible because no system dialect subsumes all others – not to mention the political problems such a standardization would induce. Instead, MitM uses a central mathematical ontology that provides an independent mediating language, via which all participating systems are aligned. All mathematical knowledge shared between the systems and exposed to the high-level VRE user is expressed using the vocabulary of this ontology. Crucially, while every system dialect makes implementation-driven, system-specific design choices, the MitM ontology can remain close to the knowledge published in the mathematical literature, which already serves as an informal interoperability layer.

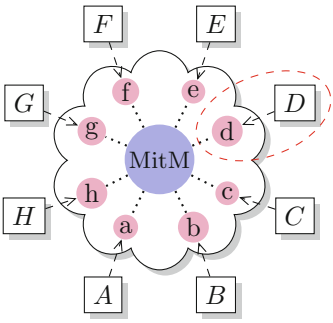


Fig. 1. MitM paradigm

The following sections describe the three components of the MitM paradigm in more detail.

2.1 The MitM Ontology

In the center, we have the **MitM Ontology**, which is a formalization of the mathematical knowledge behind the systems A to H. As a formalization framework, it uses the OMDoc/MMT format [Koh06,RK13,MMT], which was designed with this specific application in mind. We do not go into the details of OMDoc/MMT here – for our purposes, it suffices to assume that an OMDoc/MMT theory graph formalizes a language for mathematical objects as a set of typed symbols with a (formal or informal) specification of their semantics.

For example, the MitM-symbol `PolynomialRing` takes a ring r of coefficients and a number n of variables and returns the ring $r[X_1, \ldots, X_n]$ of polynomials.

Note that the purpose of the MitM ontology is not the formal verification of mathematical theorems (as for most existing formalizations of group theory), but to act as a pivot point for integrating systems. This means that it can be much nearer to the informal but rigorous presentation of mathematical knowledge in the literature. While each system dialect makes compromises and optimizations needed for a particular application domain, the MitM ontology follows the existing and already informally standardized mathematical knowledge and can thus serve as a standard interface layer between systems.

Importantly, the MitM ontology does not have to include any definitions[2] or proofs – it only has to declare the types of all relevant symbols and state (but not prove) the relevant theorems. This makes it possible for users like Jane to extend the MitM ontology quickly whereas extending formalizations usually requires extensive efforts by specialists.

2.2 Specifying System Dialects

System Dialects. It is unavoidable that each system induces its own language for mathematical objects. This is the cause of much incompatibility because even subtle differences make naive integration impossible. Moreover, due to the difficulty of the involved mathematics and the effort of maintaining the implementations, such differences are aplenty.

Fortunately, we can at least easily abstract from the user-facing surface syntax of these languages: scalable interoperability can anyway only be achieved by acting on the internal data structures of the systems. Thus, only the much simpler internal abstract syntax needs to be considered.

The symbols that build the abstract syntax trees can be split into two kinds: **constructors** build primitive objects without involving computation, and **operations** compute objects from other objects (including predicates, which we see as operations that return booleans). For purposes of interoperability it is desirable to abstract from this distinction and consider both as typed symbols. This abstraction is important because systems often disagree on the choice of constructors. Thus, we can represent the interfaces of the systems A to H as OMDoc/MMT theory graphs a to h that declare the constructors and operations (but omit all implementations of the operations) of the respective system.

Given the theory graph a representing the system dialect of A, we can express all objects in the language of system A as OMDoc/MMT objects using the symbols of a. We refer to these objects as A-objects. It is conceptually straightforward to write (or even automatically generate) the theory graph a and to implement a serializer and parser for A-objects as a part of A.[3] This is because no consideration of interoperability and thus no communication with the developers of other systems is needed.

[2] Of course, definitions are one possible way to specify the semantics of MitM-symbols.

[3] However, as we see below, this may still be surprisingly difficult in practice.

Alignments with the Ontology. The above reduces the interoperability problem to relating each system dialect to the MitM ontology. Each system dialect overlaps with the language of the ontology, but no system implements all ontology symbols and every system implements idiosyncratic operations that are not useful as a part of the ontology. Therefore, some system dialect symbols are related to corresponding symbols in the MitM ontology. We use these symbols of the MitM ontology as an intermediate representation to bridge between any two systems, e.g., by translating A-objects to the corresponding ontology objects and then those to the corresponding B-objects.

However, even when A and B deal with the "same mathematical objects", these may be constructed and represented differently, e.g., symbols can differ in name, argument order/number, types, etc. A major difficulty for system interoperability is correctly handling these subtle differences. To formalize the details of this relation, [Mül+17b] introduced OMDoc/MMT **alignments**. Technically, these are pairs of OMDoc/MMT symbol identifiers decorated by a set of key-value pairs. The alignments of a-symbols with the MitM ontology determine which A-objects correspond to MitM-objects.

The alignment of a-symbols to ontology symbols must be spelled out manually. But this is usually straightforward and easy even for inexperienced users. For example, the following line aligns GAP's symbol IsCyclic (in the file lib/grp.gd) with the corresponding symbol cyclic in the MitM ontology. The key-value pairs are used to signify that this alignment is part of a group of alignments called "VRE" and can be used for translations in both directions.

```
gap:/lib?grp?IsCyclic  mitm:/smglom/algebra?group?cylic
    direction="both" type="VRE"
```

Thus we can reduce the problem of interfacing n systems to (i) curating the MitM ontology for the joint mathematical domain, (ii) generating n theory graphs for the system dialects, (iii) maintaining n collections of alignments with the MitM ontology.

Alignments form an independent part of the MitM interoperability infrastructure. Incidentally, they obey a separate development schedule: the MitM ontology is developed by the community as a whole as the understanding of a mathematical domain changes. The system dialects are released together with the systems according to their respective development cycle. The alignments bridge between them and have to mediate these cycles.

2.3 MitM-Based Distributed Computation

The final missing piece for a system interoperability layer for a VRE toolkit is a practical way of transporting objects between systems. This requires two steps.

Firstly, if the system dialects and alignments are known, we can automatically translate A-objects to B-objects in two steps: A to ontology and ontology to B. This two-step translation has been implemented in [Mül+17a] based on the MMT

system [Rab13,MMT], which implements the OMDoc/MMT format along with logical and knowledge management algorithms.

Secondly, each system A has to be able to serialize/parse A-objects and to send them to/receive them from MMT. In the OpenDreamKit project we use the OpenMath SCSCP (Symbolic Computation Software Composability) protocol [Fre+] for that. It is straightforward to extend a parser/serializer for A-objects to an SCSCP clients/server by implementing the SCSCP protocol on top of, e.g., sockets or using an existing SCSCP library.

3 The MitM Ontology for Computational Group Theory

Jane's use case involves groups and actions, polynomials, rings and ideals, and Gröbner bases, all of which must be formalized in the MitM ontology. Due to space restrictions, we only describe the ontology for computational group theory (CGT) as an example. This formalization can be found at [Mitb].

3.1 Type Theory and Logic

OMDoc/MMT formalizations must be relative to foundational logic, which is itself formalized in OMDoc/MMT. As foundation for all formalizations in MitM [Mita], we use a polymorphic dependently typed λ-calculus with two universes type and kind (roughly analogous to sets and proper classes in set theory) and subtyping. It provides dependent function types {a:A}B(a), representing the type of all functions mapping an argument a:A to some element of type B(a). If B does not depend on the argument a, we obtain the simple function type A→B.

For formulas, we use a type prop and a higher order logic where quantifiers range over any type. We furthermore follow the judgments-as-type paradigm by declaring a function ⊢:prop→type mapping propositions to the **type of their proofs**, which allows us to declare proof rules as functions mapping proofs (of the premises) to a proof (of the conclusion).

The judgment A<:B expresses that A is a subtype of B. We use power types (the type of subtypes of a type) and predicate subtyping {'a:A | P(a)'}. The latter makes type-checking undecidable, but that is necessary for natural formalizations in many areas of mathematics.

Additionally we extend our type theory with record types, which is critical for formalizing mathematical structures. In particular, ModelsOf T is the record type of models of the theory T. This lets us, e.g., define groups by the theory of operations and its signature and axioms, while group = ModelsOf group_theory is the type of all models of said theory, i.e., all groups, as seen in Fig. 2. Any element g:group thus represents an actual group, whose operations and axioms can

```
theory group : base:?Logic =
  theory group_theory : base:?Logic =
    include ?monoid/monoid_theory

    inverse : U → U # 1 ⁻¹ prec 24
    inverseproperty : ⊢ ∀ [x] x ∘ x ⁻¹ ≐ e
  group = ModelsOf group_theory
```

Fig. 2. MitM ontology fragment

be accessed via record field projections (e.g. g.inverse yields the inverse operation of g. Since axioms are turned into record type fields as well, actually constructing a record of type group corresponds to proving that the field universe and the operations provided in the record do in fact form a group.

3.2 Group Theory

Our formalization of CGT follows the template of its implementation in GAP, and requires different levels of abstraction – currently *abstract*, *representation*, *implementation*, and *concrete*. From our experience, we expect this pattern to be applicable across computational algebra, possibly with additional levels of abstraction. The left box in Fig. 3 shows the levels and their relation to the constructors and operations of GAP.

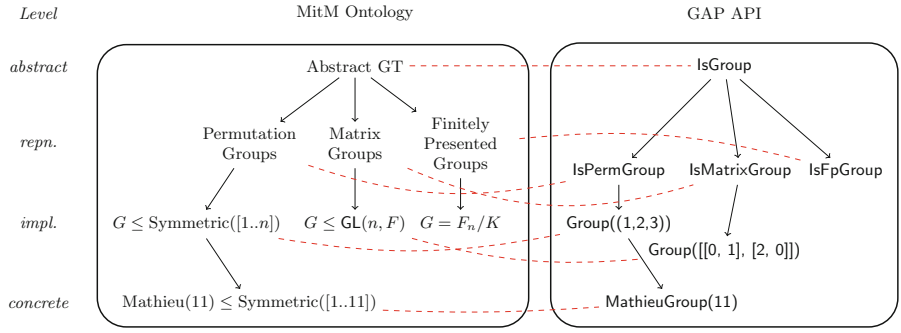


Fig. 3. Alignments between the MitM ontology and the GAP API

Abstract Level. This contains the theory of *Groups*: the group axioms, generating sets, homomorphisms, group actions, stabilisers, and orbits. This also easily leads into definitions of centralisers – i.e. stabilisers of elements under conjugation – and normalisers – i.e. stabilisers of subgroups under conjugation, stabiliser chains, Sylow-p subgroups, Hall subgroups, and many other concepts.

OMDoc/MMT also allows expressing that there are different equivalent definitions of a concept: We defined group actions in two ways and used *views* to express their equivalence.

Representation Level. Abstract groups are represented in different ways as concrete objects suitable for computation: as groups of permutations, groups of matrices, finitely presented groups, algebraic constructions of groups, or using polycyclic presentations.

Many representations arise naturally from *group actions*: If we are considering symmetry in a setting where we want to apply group theory, we start with a group action, for example a group acting on a graph by permuting its vertices.

The universal tool to bridge the gap between groups, representations and canonical representatives are *group homomorphisms*, particularly embeddings and isomorphisms, which are used extensively in GAP. This is reflected in our approach.

Implementation Level. At this level we encode implementation details: Permutation groups in GAP are considered as finite subgroups of the group $S_{\mathbb{N}+}$, and defined by providing a set of generating permutations. GAP then computes a stabiliser chain for a group that was defined this way, and naturally considers the group to be a subgroup of $S_{[1\ldots n]}$, where n is the largest point moved.

Concrete Level. It is at the concrete level where the computation happens: while the higher levels are suitable for mathematical deduction and inference, this level is where GAP (or any other system providing computational group theory) does its work. If a group (or a group action) has been constructed by giving generators through MitM, GAP can now compute the size of the group, its isomorphism type, and perform all the other operations that are available via the GAP system dialect.

4 The System Dialects of GAP, SageMath, and Singular

We now show how we produce OMDoc/MMT theory graphs that specify the system dialects of GAP, Singular, and SageMath. The three systems are sufficiently different that we can consider the development presented in this section a meaningful case study in the methodology and difficulty of exposing the APIs of real-world systems as of formally described system dialects.

In each case, we had to overcome major implementation difficulties and invest significant manpower. In fact, even the serialization of internal abstract syntax trees as OMDoc/MMT objects proved difficult, for different system-specific reasons. In the following, we summarize these efforts.

4.1 SageMath

We first consider our previous work [Deh+16] regarding a direct (i.e., without MitM) integration of SageMath and GAP. Here SageMath's native interface to GAP is upgraded from the **handle paradigm** to the **semantic handles** paradigm. In the former, when a system A delegates a calculation to a system B, the result r of the calculation is not converted to a native A object (unless it is of some basic type); instead B just returns a handle h (i.e., some kind of reference) to the B-object r. Later, A can run further calculations with r by passing it as argument to functions or methods implemented by B. Additionally, with a **semantic** handle, h behaves in A as if it was a native A object. In other words, one adapts the API satisfied by r in B to match the API for the same kind of objects in A. For example, the method call `h.cardinality()` on a SageMath handle `h` to a GAP group `G` triggers in GAP the corresponding function call `Size(G)`.

This approach avoids the overhead of back and forth conversions between A and B and enables the manipulation of B-objects from A even if they have no native representation in A. However, if these B-objects need to be acted on by native operations of A or other systems (as in Jane's scenario), we actually have to convert the objects r between A and B.

API. In [Deh+16] we describe the extraction of some of SageMath's API from its **categories**. This exploited the mathematical knowledge explicitly embedded in the code to cover a fairly large area of mathematics (hundreds of kinds of algebraic structures such as groups, algebras, fields, ...), with little additional efforts or need to curate the output. This extraction did not cover the constructors, knowledge about which is critical for (de)serialization, nor other areas of mathematics (graph theory, elliptic curves, ...) where SageMath developers currently do not use categories (usually because the involved hierarchies of abstract classes are shallow and easily maintained by hand).

To extract more APIs, we took the following approach:

1. We constructed a list of typical SageMath objects.
2. We used introspection to analyze those objects, crawling recursively through their hierarchy of classes to extract constructors and available methods together with some mathematical knowledge.

At this stage, the list of objects was crafted by hand to cover Jane's scenarios and some others. In a later stage, we plan to take advantage of one of SageMath's coding standards: every concrete type must be instantiated at least once in SageMath's tests and the instance passed trough a generic test suite that runs sanity checks for its advertised properties (e.g. associativity, ...). Therefore, by a simple instrumentation of SageMath's test framework, we could run our exporter on a fairly complete collection of SageMath objects.

The process remains brittle and the export will eventually require much curation:

- The signature of methods is incomplete: it specifies the number and names of the arguments, but only the type of the first argument.
- For constructors, the type of all the arguments is known, but only for the specific call that led to the construction of the introspected object.
- There is no distinction between mathematically relevant methods and purely technical ones like data structure manipulation helpers.
- The export is very large and seems of limited use without alignments with the MitM ontology. At this stage we do not foresee much opportunities to produce such alignments other than manually.

Nonetheless, we consider this an important first step toward fully automatic extraction of the SageMath API. Moreover, we expect further improvements by code annotations in SageMath (e.g., the ongoing porting of SageMath from Python 2 to Python3 will enable **gradual typing**, which we hope to become widely adopted by the community) or using type inference in SageMath and/or MitM.

Serialization and Deserialization. Because SageMath is based on Python, it benefits from its native serialization support. For example, the dihedral group D_4 is serialized as a binary string, which encodes the following straight line program to be executed upon deserialization:

```
pg_unreduce = unpickle_global('sage.structure.unique_representation', 'unreduce')
pg_DihedralGroup =
    unpickle_global('sage.groups.perm_gps.permgroup_named', 'DihedralGroup')
pg_make_integer = unpickle_global('sage.rings.integer', 'make_integer')
pg_unreduce(pg_DihedralGroup, (pg_make_integer('4'),), {})
```

The first three lines recover the constructors for integers and for dihedral groups from SageMath's library. The last line applies them to construct successively the integer 4 and D_4.

Up to concrete syntax, this serialization is already close to the desired SageMath system dialect. We can therefore extend Python's native (de)serializer to use OMDoc/MMT as an alternative serialization format (using the Python library [POMa]). This has the advantage of using optimizations implemented in Python's serialization, e.g., structure sharing for identical subexpressions.

Still, systematically expanding OMDoc/MMT serialization to the *entire* SageMath library requires significant manpower and can only be a long-term goal. To increase community support, our design elegantly decouples the problem into (i) instrumenting the serialization to generate OMDoc/MMT as an alternative target format, and (ii) structural improvements of the serialization that benefit SageMath in general.

In particular, our serialization of SageMath objects is **by construction** rather than **by representation**, i.e., we serialize the constructor call that was used to build an object instead of the low-level Python representation of the resulting object. This is important to hide implementation details and allow for straightforward alignments. From the origin, the SageMath community has internally promoted good support for serialization as this is a fundamental building block for communication between parallel processes, databases, etc. Thus, it already values serialization by construction as superior because it is usually more concise and more robust under changes to SageMath. Therefore, independent of the purposes of this paper, we expect a synergy with the SageMath community toward improving serialization.

4.2 GAP

In [Deh+16], we already described our general approach to extract APIs from the GAP system. We have now improved on this work considerably.

Firstly, we improved the MitM foundation so that the primitives of GAP's type system can be expressed in the MitM ontology.[4] GAP's type system heavily uses subtyping: **filters** express finer and finer subtypes of the universal type IsObject. Moreover, an object in GAP can learn about its properties, meaning its type is refined at runtime: a group can learn that it is Abelian or nilpotent and change its type accordingly.

Secondly, we devised and implemented a special treatment of GAP's constructors during serialization. As GAP only has a weak notion of object construction,

[4] In the future MMT might even serve as an external type-checker for GAP.

we achieved this by manually identifying and annotating all functions that create objects in the GAP code base and then instrumenting them to store which arguments they were called with. With the constructor annotation in place, it is possible to have GAP represent any object in a running session as either a primitive type (integers, permutations, transformations, lists, floats, strings), or as a constructor applied to a list of arguments.

The instrumentation itself is minimal – 57 lines of GAP code, plus 100 lines for serializing and parsing. The main – and indeed considerable – challenge was to identify the constructors and their arguments. In GAP, objects are created by calling the function Objectify with a type and some arguments. Hence we analyzed all call-sites to this function and some light inference of the enclosing function. This amounted to 665 call sites in the GAP library and an additional 1664 in the standard package distribution. The instrumentation will be released as part of a future version of GAP, making GAP fully MitM capable.

As a major positive side-effect of our work, this instrumentation led to general improvements of the type infrastructure in GAP. For example, it enables static type analysis, which can be used to optimize the dynamic method dispatch and thus hopefully lead to efficiency gains in the system.

4.3 Singular

As we only need a very small part of Singular for our case study, we were able to use the existing OpenMath content dictionaries for polynomials [OMCP] as the Singular system dialect. These are part of a standard group of content dictionaries that describe (some) mathematical objects at a high level of abstraction to be universally applicable. OMDoc/MMT understands OpenMath, i.e., it can use these content dictionaries as OMDoc/MMT theories.

Building on the OpenMath toolkits for OpenMath phrasebooks [POMa] and SCSCP communication [POMb] in Python – which were developed for SageMath in the OpenDreamKit project, we wrapped Singular in a thin layer of Python code that provides SCSCP communication. This work was undertaken by the sixth author as part of a summer internship in about a week without prior expert knowledge of the system. Of course, if we want to achieve a more comprehensive coverage of the Singular dialect, we will have to either manually write a theory graph or instrument Singular for extraction as we did for SageMath or GAP above.

4.4 Alignments

Finally we have to curate the alignments between the system dialects and the MitM ontology. These alignments are currently produced and curated manually using the approach, repository, and syntax described in [Mül+17b,Mül+17a]. In the future, we will also consider automatically extracting alignments from the existing ad-hoc SageMath-to-X translations. These are (mainly) given as SageMath code annotations that relate SageMath operations and constructors with those of system X.

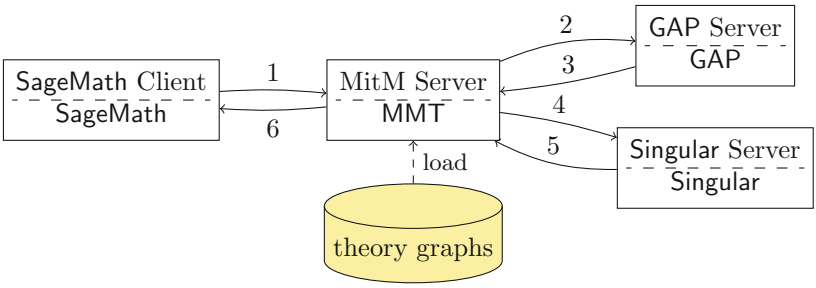


Fig. 4. MitM interaction in Jane's use case

5 Distributed Computational Group Theory

Figure 4 shows the overall architecture with an MitM server as the central mediator. All arrows represent the transfer of OMDoc/MMT objects via SCSCP. Critically, the MitM server also maintains the alignments and uses them to convert between system dialects.

We have extended the MMT system [Rab13] with an SCSCP server/client so that it can receive/send objects from/to computation systems. For the GAP server, we built on pre-existing SCSCP support. To obtain an SCSCP server for Singular, which does not have native SCSCP support, we wrapped Singular in a python script that includes the pyscscp library [POMb]. In SageMath, we directly programmed the client interface to the MitM server.

The resulting system forms the nucleus of the OpenDreamKit interoperability layer. It can already delegate computations between the three participating systems as long as the exchanged objects are covered by the MitM ontology, the alignments, and the formalizations of the system dialects.

Jane's Use Case. Initially, Jane has already built in SageMath the ring $R = \mathbb{Z}[X_1, X_2, X_3, X_4]$, the group $G = D_4$, the action A of G on R that permutes the variables, and the polynomial $p = 3 \cdot X_1 + 2 \cdot X_2$. She now calls

```
MitM.Singular(MitM.Gap.orbit(G, A, p)).Ideal().Groebner().sage()
```

which results in the following steps (the numbers on the edges of the graph of Fig. 4 indicate the order of communications when processing Jane's use case):

1. Jane uses SageMath to call the MitM server with the command above, which includes both the computation to be performed and information about which system to use at which step.
2. The MitM server translates MitM.Gap.orbit(G, A, p) to the GAP system dialect and sends it to GAP.
3. GAP returns the orbit:

$$O = [3X_1 + 2X_2, 2X_3 + 3X_4, 3X_2 + 2X_3, 3X_3 + 2X_4, \\ 2X_2 + 3X_3, 3X_1 + 2X_4, 2X_1 + 3X_4, 2X_1 + 3X_2].$$

4. The MitM server translates MitM.Singular(O).Ideal().Groebner() to the Singular system dialect and sends it to Singular.
5. Singular returns the Gröbner base B.
6. The MitM server translates B to the SageMath system dialect and sends it to SageMath, where the result is shown to Jane.

$$B = [X_1 - X_4, X_2 - X_4, X_3 - X_4, 5 * X_4].$$

Alternative Use Case. Suppose Jon, one of Jane's colleagues, prefers working in GAP, and he wants to compute the Galois group of the rational polynomial $p = x^5 - 2$. He discovers the GAP package `radiroot`, which promises this functionality, but unfortunately the package does not work for this polynomial and thus GAP alone cannot solve Jon's problem.

Jon hears from Jane that he should use SageMath, because she knows it can compute Galois groups. So, from GAP, he calls

```
G := MitM("Sage", "GaloisGroup",p)
```

which gives him the desired Galois group as a GAP permutation group. Having heard of Jane's experiments, he can further run her orbit and Gröbner basis calculation starting from this new group, without leaving his favorite computing environment.

Finally, Jon, being a proficient GAP user, also knows that he can now install a **method** in GAP by calling

```
InstallMethod(GaloisGroup, "for a polynomial", [IsUnivariatePolynomial],
              p -> MitM("Sage", "GaloisGroup", p))
```

that will compute the Galois group of any rational polynomial transparently for him whenever he calls GaloisGroup for a rational polynomial in GAP. And thus (at the price of using multiple systems) a significant part of the 1800-line radiroot package can be replaced by a few lines in GAP, taking advantage of the work of the SageMath community and participating in any future improvements of SageMath. In fact, Sage itself delegates to the PARI system – another one of the OpenDreamKit systems – for this computation. So in the future GAP might directly delegate to PARI instead, bypassing the need of iterated translations.

6 Conclusion

We have implemented the MitM approach to integrating mathematical software system based on formalizations of the underlying mathematical knowledge. The main investment here was the curation of an MitM Ontology, the generation of formal specifications of system APIs for SageMath, GAP, and Singular, identifying the alignments of these APIs with the ontology, implementing an MitM server that can use alignments to translate between systems, and implementing the SCSCP protocol for all involved systems.

Our case study showed that MitM-based integration is an achievable goal. Delegation-based workflows can either be programmed directly or embedded into the interaction language of the mathematical software systems.

The main advantages and challenges claimed by the MitM framework come from its loosely coupled and knowledge-based nature. Compared to ad-hoc translations, MitM-based interoperability is relatively expensive as objects have to be serialized into (possibly large) OMDoc/MMT objects, transferred via SCSCP to MMT, parsed, translated into another system dialect, serialized and transferred, and parsed again. On the other hand, instead of implementing and maintaining n^2 translations, we only have to establish and maintain n collections of system APIs and their alignments to the MitM ontology. This makes the management of interoperability much more tractable:

1. The MitM ontology is developed and maintained as a shared resource by the community. We expect it to be well-maintained, since it can directly be used as a documentation of the functionality of the respective systems.
2. All the workflows are star-shaped: instead of requiring expert knowledge in two systems – a rare commodity even in open-source projects, and even for the system experts involved in this paper – and keeping up with their changes, the MitM approach only needs expertise and change management for single systems.

All in all, these translate into a "business model" for MitM-based cooperation in terms of the necessary investment and achievable results, which is based on the well-known *network effects*: the joining costs are in the size of the respective system, whereas the rewards – i.e. the functionality available by delegation – is in the size of the network.

This network effect can be enhanced by technical refinements we are currently studying: For instance, if we annotate alignments with a "priority" value that specifies how canonically/efficiently/powerfully a given system implements a given MitM operation, then we can let the MMT mediator automatically choose a suitable target system for a requested computation (as opposed to our current setup where Jane specifies which systems she wants to use). On the other hand, for workflows where we do not need or want service-discovery, alignments can be "compiled" into n^2 transport-efficient direct translations that may even eliminate the need for serialization and parsing.

Acknowledgements. The authors gratefully acknowledge the fruitful discussions with other participants of work package WP6, in particular Alexander Konovalov on SCSCP, Paul Dehaye on the SageMath export and the organization of the MitM ontology, and Luca de Feo on OpenMath phrasebooks and the SCSCP library in python. We acknowledge financial support from the OpenDreamKit Horizon 2020 European Research Infrastructures project (#676541) and DFG project RA-18723-1 OAF.

References

[Cre16] Cremona, J.: The L-functions and modular forms database project. Found. Comput. Math. **16**(6), 1541–1553 (2016). https://doi.org/10.1007/s10208-016-9306-z

[Deh+16] Dehaye, P.-O., et al.: Interoperability in the OpenDreamKit project: the Math-in-the-Middle approach. In: Kohlhase, M., Johansson, M., Miller, B., de de Moura, L., Tompa, F. (eds.) CICM 2016. LNCS (LNAI), vol. 9791, pp. 117–131. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42547-4_9. https://github.com/OpenDreamKit/OpenDreamKit/blob/master/WP6/CICM2016/published.pdf

[Fre+] Freundt, S., et al.: Symbolic computation software composability protocol (SCSCP). Version 1.3. https://github.com/OpenMath/scscp/blob/master/revisions/SCSCP_1_3.pdf. Accessed 27 Aug 2017

[GAP] The GAP Group: GAP - groups, algorithms, and programming. http://www.gap-system.org. Accessed 30 Aug 2016

[Jup] Project Jupyter. http://www.jupyter.org. Accessed 22 Aug 2017

[Koh+11] Kohlhase, M., et al.: The planetary system: Web 3.0 & active documents for STEM. Procedia Comput. Sci. **4**, 598–607 (2011). https://doi.org/10.1016/j.procs.2011.04.063. Sato, M., et al. (eds.) Special issue: Proceedings of the International Conference on Com- putational Science (ICCS). Finalist at the Executable Paper Grand Challenge

[Koh06] Kohlhase, M.: OMDoc – An Open Markup Format for Mathematical Documents [Version 1.2]. LNCS (LNAI), vol. 4180. Springer, Heidelberg (2006). https://doi.org/10.1007/11826095. http://omdoc.org/pubs/omdoc1.2.pdf

[LMFDB] The LMFDB Collaboration: The L-functions and modular forms database. http://www.lmfdb.org. Accessed 02 Jan 2016

[Mita] MitM/Foundation. https://gl.mathhub.info/MitM/Foundation. Accessed 01 Sept 2017

[Mitb] MitM/Groups. https://gl.mathhub.info/MitM/groups. Accessed 01 Sept 2017

[MMT] MMT - Language and System for the Uniform Representation of Knowledge. Project web site. https://uniformal.github.io/. Accessed 30 Aug 2016

[Mül+17a] Müller, D., et al.: Alignment-based translations across formal systems using interface theories. In: Fifth Workshop on Proof eXchange for Theorem Proving - PxTP 2017 (2017). http://jazzpirate.com/Math/AlignmentTranslation.pdf

[Mül+17b] Müller, D., Gauthier, T., Kaliszyk, C., Kohlhase, M., Rabe, F.: Classification of alignments between concepts of formal mathematical systems. In: Geuvers, H., England, M., Hasan, O., Rabe, F., Teschke, O. (eds.) CICM 2017. LNCS (LNAI), vol. 10383, pp. 83–98. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62075-6_7

[OMCP] OpenMath CD Group: polygrp. http://www.openmath.org/cdgroups/polygrp.html. Accessed 01 Sept 2017

[POMa] An OpenMath 2.0 implementation in Python. https://github.com/OpenMath/py-openmath. Accessed 04 Sept 2016

[POMb] An SCSCP module for Python. https://github.com/OpenMath/py-scscp. Accessed 04 Sept 2016

[Rab13] Rabe, F.: The MMT API: a generic MKM system. In: Carette, J., Aspinall, D., Lange, C., Sojka, P., Windsteiger, W. (eds.) CICM 2013. LNCS (LNAI), vol. 7961, pp. 339–343. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39320-4_25

[RK13] Rabe, F., Kohlhase, M.: A scalable module system. Inf. Comput. **230**, 1–54 (2013). http://kwarc.info/frabe/Research/mmt.pdf

[Sage] The Sage Developers: SageMath, the sage mathematics software system. http://www.sagemath.org. Accessed 30 Sept 2016

[SNG] Singular. https://www.singular.uni-kl.de/. Accessed 22 Aug 2017

On Interval Methods with Zero Rewriting and Exact Geometric Computation

Stefan Schirra(✉) and Martin Wilhelm

Department of Simulation and Graphics, Faculty of Computer Science, University of Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany
{stschirr,mwilhelm}@isg.cs.uni-magdeburg.de

Abstract. We oppose interval-symbol methods with zero rewriting developed by Shirayanagi and Sekigawa [14,31–33] to the exact geometric computation paradigm [17,37], especially to exact decisions computation via lazy adaptive evaluation with expression-dags, in doing so carving out analogies and disparities.

Keywords: Interval-symbol method · Exact geometric computation
Robustness and precision issues · Verified numerical computing

1 Introduction

In a sequence of papers Shirayanagi and Sekigawa [14,31–33] propose so-called *interval methods with zero rewriting* to get exact results with bigfloat interval arithmetic in order to avoid expensive exact arithmetic. They propose to use bigfloat interval arithmetic of a fixed precision and to replace any zero-containing interval I arising during computation by the zero interval, i.e., the point interval representing zero. This is called *zero rewriting*. So the rule is "if we don't know the sign for sure, we assume it is zero". If the result of the computation cannot be verified, computation is repeatedly started over with increased bigfloat precision. Their goal of using interval methods with zero rewriting is getting both exact numerical and exact combinatorial results more efficiently.

Intuitively, interval methods with zero rewriting give us correct results if the underlying bigfloat precision is sufficiently high: During a finite computation we compute only finitely many numerical values. If our precision suffices to let the bigfloat interval arithmetic separate the non-zero values from zero, all zero rewritings are correct. This observation is made formal by Sweedler and Shirayangi in their stabilization theorem [28,29]. However, usually we don't know the sufficient precision in advance. Therefore, Shirayanagi and co-workers suggest to repeatedly increase the precision until correctness can be verified. In Sect. 3 we recap the various intervals methods with zero rewriting as well as the corresponding approaches to result verification.

In contrast to the interval methods with zero rewriting, the primary motivation for the exact geometric computation paradigm [37] is achieving numerical

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 211–226, 2017.
https://doi.org/10.1007/978-3-319-72453-9_15

robustness by avoiding inconsistent decisions: If all evaluations of geometric predicates are exact, inconsistencies are a non-issue. The emphasis is on getting exact combinatorial results; numerical data need not be exact. In this paper we compare interval methods with zero rewriting to techniques that are used in computational geometry for implementing the exact geometric computation paradigm, in particular to lazy adaptive evaluation with expression-dags, a general purpose approach for exact geometric computation with real algebraic numbers.

Shirayanagi and his co-workers were initially interested in algebraic algorithms only, especially in reducing the cost of algorithms manipulating polynomials by using floating-point arithmetic [30]. In polynomial algebra exactness of the coefficients of the computed polynomials is inevitable. The situation is somewhat different in computational geometry where exactness of the numerical part of the output is usually much less important than exactness of the combinatorial part. Often, together with the numerical data in the input, the combinatorial part serves as an exact symbolic representation of the numerical data in the output. For example, knowing the sites giving rise to a Voronoi vertex suffices to recover its numerical coordinates exactly.

In [33], Shirayanagi and Sekigawa apply their approach to geometric computing, more precisely, to planar convex hull computation. Are interval-symbol methods with correct zero rewriting a valuable alternative to the techniques used to implement the exact geometric computation paradigm? We discuss this question by comparing different versions of interval methods with zero rewriting to techniques used in computational geometry to implement the exact geometric computation paradigm. Our comparative discussion is accompanied by experiments on convex hull computation in the plane.

2 Exact Geometric Computation Paradigm

Let us call a representation of a real number r exact if we can read-off the sign of r directly without any further computation and the representation allows us to approximate r to any accuracy we want. While exact arithmetic always maintains exact representations, exact decisions computation ensures only that all comparisons of numerical values as well as all sign computations are correct. Note that exact values are not always necessary (actually, as it turns out, in our context, they are never necessary) for exact decisions, sufficiently close approximations are adequate. The exact geometric computation paradigm advocated by Yap [37] extends this relaxation from the arithmetic level to the geometric level. It asks for exact results of geometric predicates only. The error-free evaluation of predicates makes robustness problems a non-issue in geometric computing. Since exact geometric computation allows for incorrect decisions within the evaluation of geometric predicates, it is less strict than exact decisions computation. Of course, computation with exact arithmetic subsumes exact decisions computation and exact geometric computation can be implemented by exact decisions computation. The latter approach is sometimes called exact decisions geometric computation as well. Structural filtering [9] carries the relaxation even further.

It asks for exact substeps only. Intermediate inexactness is permitted and repaired at the end of the execution of a substep where necessary. A substep might involve several predicates or might even embrace a whole algorithm, depending on the application.

The precision and robustness problem is closely related to the problem of degeneracies in geometric computing. On the one hand, exact decisions computation is required to detect and handle degeneracies exactly, on the other hand, exact decisions computation is a prerequisite for symbolic perturbation methods that allow one to circumvent the handling of degeneracies. Avoiding the handling of degeneracies is the goal of the topology-oriented approaches by Sugihara and his co-workers [34–36] and of controlled perturbation [10–12,18]. Both approaches try to get along with floating-point arithmetic. Designing such algorithms seems to be more difficult and overall, compared to the exact geometric computation paradigm, these approaches still seem to be less mature or less applicable in general. And of course, this way we can get approximate solutions only.

Over the years effective techniques [38] have been used and developed to support the efficient implementation of the exact geometric computation paradigm, most notably floating-point filters [7] and approaches exploiting error-free floating-point transformations [27]. A general purpose approach is coupling lazy adaptive evaluation with expression dags [1]. This can be done both on the arithmetic level [1,3,13] and on the geometric level [8,23]. Main ingredients of these approaches are approximate expression evaluation, arithmetic filters, and constructive zero separation bounds, where a zero separation bound for an arithmetic expression E is a positive real number $sep(E)$ which is a lower bound on the absolute value of E. For expressions involving operations $+, -, \cdot, /$ and $\sqrt[k]{}$ and integer (or rational) operands, zero separation bounds can be computed inductively according to the structure of an expression [4,6,19,24–26]. Zero separation bounds allow us to verify that an expression is zero if we have a sufficiently close approximation: If the sum of the absolute value of the approximation and the error bound are smaller than the zero separation bound we may conclude that the actual value is zero. The exact geometric computation paradigm is implemented in the C++-software libraries CGAL [5] and LEDA [16].

Lazy Adaptive Evaluation with Expression-Dags. In the sequel we focus on arithmetic expression-dags since they are more closely related to the interval with symbols approach by Shirayanagi and Sekigawa. Recording the computation history of numerical values in expression-dags, i.e., expression-"trees" that may share common subexpressions, allows one to (re)compute an approximation of the value of the expression at any time at any accuracy. Using such dags we can adaptively compute the sign of an expression correctly by repeatedly increasing the precision until the error is less than the absolute value or constructive zero separation bounds allow us to conclude that the actual value is zero. Furthermore, we apply lazy evaluation to sign computation, i.e., sign computation is delayed until the sign is actually needed. This strategy is implemented in C++-number types in CORE [13], leda::real [3], and RealAlgebraic [20]. Since all

sign computations and hence all decisions in geometric predicates are exact, we banish inconsistencies caused by numerical imprecision.

3 Variants of Interval Methods with Zero Rewriting

Over the years Shirayanagi and Sekigawa and Shirayanagi and Katayama came up with different versions of interval methods with zero rewriting.

Interval Method with Zero Rewriting. In the simplest version, Shirayanagi and Sekigawa [32] propose to replace every zero-containing interval I arising during bigfloat interval computation with a certain precision by the zero interval immediately, without any verification of this step, and to verify the result of the overall computation afterwards. If this verification fails the whole computation is re-done with increased precision of bigfloat interval arithmetic. This is repeated until verification succeeds, see Fig. 1(a).

Zero rewriting reminds one of epsilon tolerancing, also called epsilon tweaking, where we replace tests for zero by comparisons of absolute values with certain epsilons. In contrast to zero rewriting where we get "sound epsilons" through interval arithmetic, finding epsilons for epsilon tolerancing is often guess work and "an art that requires infinite patience" [22]. Usually, people applying epsilon tweaking do not attempt to verify results. Of course, epsilon tweaking does not implement the exact geometric computation paradigm. Unfortunately, like epsilon tweaking, zero rewriting does not abolish inconsistencies. For example, equality testing is not transitive. We might rewrite the distance between a and b to zero as well as the distance between b and c, but not the distance between a and c. Since inconsistent decisions can still arise, epsilon tweaking is not a recommended approach to precision and robustness problems in computational geometry.

Correcting afterwards presumes that the algorithm runs until the end regardless of whether zero rewriting is correct. Therefore, interval method with plain zero rewriting can be used only with (quasi-)robust algorithms which always compute some (kind of) useful output, which can be verified afterwards. However, designing a robust geometric algorithm is a difficult task. Since many geometric algorithms are not (quasi-)robust, applicability of the initial version of interval methods with zero rewriting is rather limited. The idea of correcting afterwards is present in *structural filtering* approaches in computational geometry, too. While Shirayanagi and Sekigawa suggest to simply rerun the algorithm with higher precision, structural filtering aims at repairing the result of a structurally filtered step using other exact methods. Kettner and Welzl [15] apply the idea to convex hull computation in the plane. Funke et al. discuss the above robustness problem for structural filtering at the algorithm level in [9].

Zero rewriting takes place immediately when a new zero-containing bigfloat interval is created. One could think of lazy zero rewriting as well, where zero rewriting takes place only if we ask for the sign of a numerical value approximated by a bigfloat interval.

Intervals with Symbols. Since output verification without exact numerical values might be difficult, Shirayanagi and Sekigawa [32] propose to maintain symbolic information in addition to the approximating bigfloat intervals. They store symbol strings to record the computation history of a numerical value. Keeping track of the history of a value allows one to get exact representations for the numerical values approximated by the intervals at the very end of the computation.

There are two versions presented in [32]. In a first version, the symbol strings of the operands of an arithmetic operation are copied into the symbol string of the result, together with a symbol representing the arithmetic operation performed. These symbol strings are stored with the intervals. Unfortunately, this lets the size of the symbol strings grow quickly, since the same information is stored in several places. Therefore, Shirayanagi and Sekigawa propose a second version where intervals with symbols share computation histories: They maintain a global symbol list where each entry records an operation together with list indices for the operands. Now only a list index is stored with an interval. List index and global symbol list allow one to reconstruct the computation history for a value and hence enable exact (re-)computation. Still, zero rewriting is applied whenever a zero-containing interval arises. No attempt is made yet to verify that the actual value is zero.

Recording computation history enables reconstruction of exact values and eases verification of the computed result afterwards. As before, if verification of the computed output fails, computation is restarted from scratch with higher precision bigfloat interval arithmetic until we obtain a correct result. As discussed above output verification must be possible somehow. If we have a selective geometric problem, where all numerical data of the output is present in the input already, there is no need to verify these numerical data and inspection of the combinatorial part suffices. If we have a constructive geometric problem, i.e., new numerical data is constructed, we can make use of the symbolic information stored with the intervals and the symbol list to verify these data. However, according to [32], the symbolic part is erased when zero rewriting takes place. Unfortunately this way we lose the option to check zero values for correctness at the end of computation and it is unclear how exactness is ensured in such a case.

Maintenance of computation history in a global symbol list is similar to maintenance of computation history in expression dags. Copying is avoided and common subexpressions are shared. However, while expression dags use reference counting to detect when subexpressions are not needed anymore, there is no corresponding mechanism in the global symbol list. The symbol list then contains information for many numerical values that are not existent anymore. Therefore, global symbol list grows continuously and can become quite large even for flat computations which are typical for most algorithms for low-dimensional geometric problems.

Interval-Symbol Method with Correct Zero Rewriting. The use of symbols also allows for the verification of zero rewritings [32]. This verification is

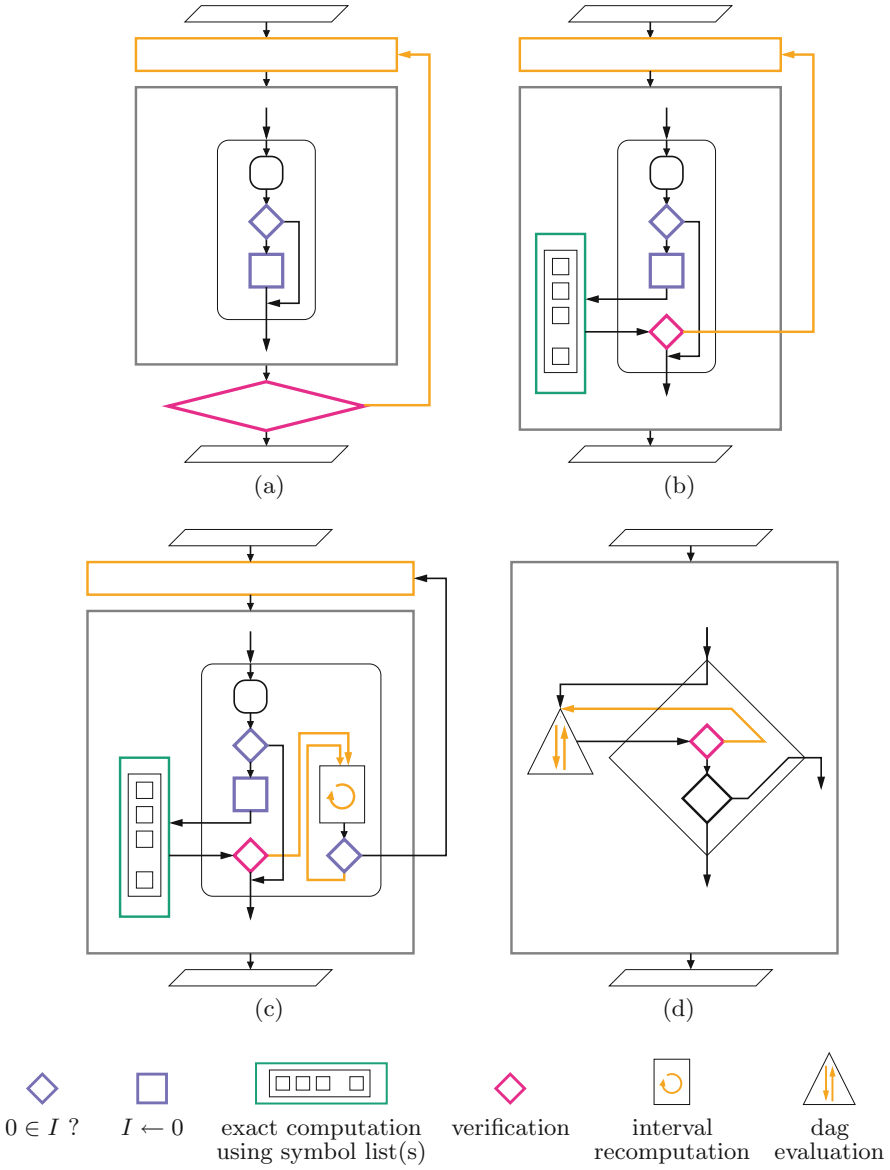


Fig. 1. Pictorial outlines of (a) simple plain interval method with zero rewriting, (b) interval-symbol method with correct zero rewriting, (c) improved version of (b), and (d) lazy adaptive evaluation with expression-dags. In cases (a)–(c) we convert numerical data to bigfloat intervals with precision p before running the algorithm and restart the process with increased precision if output verification fails (a) or if we detect an incorrect zero rewriting (b), (c), where in (c) we determine the new precision based on the incorrect zero rewriting. Zero rewriting takes place when a new interval is created while lazy evaluation is postponed to decision steps.

done by an exact computation according to the computation history recorded in the symbol strings or list. The overall strategy is to restart computation from scratch with increased precision as soon as verification by exact computation fails to confirm zero rewriting, i.e., computation is re-started with increased precision if the current precision bigfloat interval arithmetic does not suffice to separate a non-zero value from zero, see Fig. 1(b).

Intervals with correct zero rewriting reminds us of floating-point filters. With a floating-point filter we try to verify non-degeneracy by fast hardware-supported floating-point arithmetic and error bounds, i.e. interval arithmetic. If verification fails, we switch to an exact computation or some other exact method. Of course, one can generalize floating-point filters to arithmetic filters using bigfloat arithmetic instead of hardware-supported floating-point arithmetic. With standard floating-point filters, whenever we have a zero-containing interval we use exact arithmetic to compute the exact sign. However, now that we know the exact sign we continue our computation with the evaluation of other geometric predicates. There is no restart from scratch. In order to compute the exact sign we must have access to the exact input data. Hence, such filters are most applicable in geometric predicates that operate on the input data directly. In cascaded computations we must have some other means for getting access to exact input data, e.g. maintaining expression dags.

Recently, Katayama and Shirayanagi [14] revised and improved the iteration strategy. Now, whenever interval arithmetic with the current precision gives us a zero-containing interval where verification fails, using the information in symbol strings or list, only the computation of this interval is rerun with repeatedly increased precision until we get an interval that does not contain zero anymore. The final sufficient precision is then used when re-staring the overall computation from scratch, see Fig. 1(c).

Besides the differences in expression-dags and symbols lists pointed out above, the strategy of lazy adaptive evaluation with expression-dags is substantially different in other aspects as well, see Fig. 1(d). With lazy adaptive evaluation on expression-dags we never restart from scratch. Moreover, verified sign computation is lazy, i.e., it takes place only if requested by the algorithm in a decision step, not as soon as a zero-containing interval arises. Furthermore, precision is increased locally only, i.e., within the subexpression whose sign we would like to know, similar to what Katayama and Shirayanagi propose. However, the precision of other interval computations is not automatically increased as well. This way we can save a lot of computation time, since we use higher precision only where we need it. Usually, approximate evaluation is precision-driven. This means, we do not use the same precision for all bigfloat interval computations everywhere, but compute precisions sufficient for the operands in order to guarantee a requested approximation error at a dag node. Thus, even for the evaluation for a single value we do not use interval arithmetic with uniform precision in lazy adaptive evaluation with expression dags. Finally, there is no verification by exact arithmetic, i.e., exact computation in the usual sense. Verification is done by constructive zero separation bounds. Of course, this gives

us the correct sign, so in this sense it is an exact computation as well. By the underlying iterative approach interval methods with correct zero rewriting throw away a lot of knowledge already gained in previous iterations, since we redo computation with higher precision and hence more expensive bigfloat arithmetic, no matter whether this higher precision is necessary or not. With lazy adaptive evaluation with expression-dags iteration is always local to the sign computation and never affects re-evaluation on a global basis.

The idea of iterative trial-and-error computation is present in controlled perturbation as well. There, current computation is stopped whenever floating-point arithmetic does not suffice to verify non-degeneracy of the current perturbed input. Then computation is re-started with a larger perturbation. Usually, the precision of the floating-point arithmetic is not increased, since software bigfloats are much more expensive than hardware-supported floats.

4 Experiments

In [33], Shirayanagi and Sekigawa use planar convex hull computation to illustrate the use of interval methods with zero rewriting and so do we. Shirayanagi and Sekigawa use a computer algebra system, more precisely, MAPLE 12, to implement their code. While a computer algebra system provides a perfect infrastructure for exact computation, implementation in `C++`, the programming language used for exact geometric computation in the software libraries CGAL [5] and LEDA [16], is somewhat more challenging. Since the infrastructure provided by CGAL and LEDA does not supply an exact arithmetic in the strong sense for real algebraic numbers, we use a symbolic representation and exact decision evaluation via `leda::real` in the verification part of zero rewriting. This works for expressions involving radicals.

We implement the interval method with symbol list and correct zero rewriting as described in Sect. 3. Additionally, we implement a variant with lazy zero rewriting which does not apply zero rewriting at construction time, but defers zero rewriting to decision steps via sign computations. Initially, we used CGAL's `Gmpfi` class for bigfloat interval arithmetic which is based on MPFI, a multiple precision interval arithmetic library based on MPFR [21]. Since `Gmpfi` does unfortunately not allow us to limit the precision to less than 53 bits, we now use `leda_bigfloat_interval`, another CGAL class which couples LEDA's bigfloat number type with `boost::numeric::interval` form Boost [2]. LEDA's bigfloat number type allows us to limit the precision to less than 53 bits.

We maintain a global symbol list to avoid storing redundant information. This allows us to encapsulate all arithmetic and zero rewriting in a `C++` number type and to use this number type together with CGAL's geometric algorithms. Together with a list of exact operands the symbol list allow us to re-compute a value using another exact number type whenever zero rewriting takes place. We use `C++` exception handling to interrupt computation whenever verification of zero rewriting fails. If this happens we increase bigfloat precision and re-start computation again after clearing exact operand and symbol list. Since constants

0 and 1 arise frequently during geometric computations with CGAL, we store and reuse symbols for these constants at the beginning of the symbol list, thereby avoiding further blow-up of the symbol list. Remember that a symbol list never shrinks during an iteration with fixed precision.

At first, we consider the convex hull experiments from [33] in our `C++`-based framework. Note that Shirayanagi and Sekigawa use decimal arithmetic and talk about decimal places when referring to precision whereas we use binary places. They consider test data in several categories, cf. [33]:

Example 1. 5000 points with coordinates (x, y), where x and y are randomly generated integers satisfying $0 \leq x, y \leq 200$.

Example 2. 5000 points with coordinates (x, y), where x and y are randomly generated integers satisfying $-200 \leq x, y \leq 200$ and $x^2 + y^2 \leq 200^2$.

Example 3. 5000 points with coordinates (x, y), where x and y are randomly generated integers satisfying $0 \leq x, y \leq 400$, $x^2 + y^2 \leq 400^2$, and $y^2 \leq 3x^2$.

Example 4. The origin $(0, 0)$ and 4999 points with coordinates (x, y), where x and y are randomly generated integers satisfying $1 \leq x, y \leq 6000$, and $\frac{9}{10} \leq \frac{x}{y} \leq 1$.

We use CGAL's random point generators to create the test data accordingly and use CGAL's default convex hull algorithm which in contrast to the MAPLE code used in [33] avoids division operations. Therefore we can use arbitrary precision integers like CGAL's `Gmpz` or `leda::integer` in the verification step of zero rewriting. Since the integers generated in the categories above are fairly small, division-free computation with `double` precision always gives correct integer result everywhere. In order to observe dependence on precision we have to use LEDA's bigfloats with binary precision limited to less than 53 bits. Note that such bigfloat computation with lower precision is somewhat more expensive than bigfloat computation with default precision 53. Besides the two variants described above, we implemented a version without symbol list, analogously to the MAPLE code made available for convex hull computation by Sekigawa. This approach defers zero rewriting to decision steps as well. In contrast to the two other variants, interval-symbol method with correct zero rewriting is not wrapped in a number type, but implemented via CGAL's traits concept for planar convex hull computation.

Since running time depends on the precision we start with, we measure and report the running time of the last successful iteration of convex hull computation only. The measured time includes the cost of conversion from `int` to our number type wrapping intervals with symbols, see Table 1.

In a second set of examples, Shirayanagi and Sekigawa [33] use irrational coordinates. Corresponding to example i above there is example $i+4$ where instead of coordinates (x, y) points we now have coordinates $(\text{sign}(x) \cdot \sqrt{|x|}, \text{sign}(y) \cdot \sqrt{|y|})$, and x and y are generated as described above for example i:

Example 5. 5000 points with coordinates $(\sqrt{x}, \sqrt{y})$, where x and y are randomly generated integers satisfying $0 \leq x, y \leq 200$.

Table 1. Convex hull with integer points for four random data sets in example classes 1 to 4. Interval method with correct zero rewriting is based on `leda_bigfloat_intervals` with verification of zero rewritings via exact integer arithmetic (`leda::integer`). Lazy adaptive evaluation is `leda::real`. Running times are for the last successful iteration only.

	ISCZ	ISCZ lazy	Without symbol list	Sufficient precision	# zero rewritings	Length of symbol list	Exact integer	Lazy adapt. expr.-dags
1.1	0.188	0.144	0.124	14	0	255326	0.008	0.001
1.2	0.172	0.136	0.124	14	0	248680	0.008	0.001
1.3	0.136	0.120	0.104	14	1	247816	0.004	0.001
1.4	0.160	0.132	0.120	15	0	254182	0.004	0.001
2.1	0.160	0.128	0.132	15	0	165348	0.004	0.001
2.2	0.188	0.152	0.124	16	0	164694	0.004	0.001
2.3	0.132	0.108	0.096	16	0	165078	0.004	0.001
2.4	0.148	0.120	0.100	14	0	167378	0.004	0.001
3.1	0.152	0.124	0.104	15	0	152204	0.004	0.001
3.2	0.120	0.100	0.084	16	0	151001	0.004	0.001
3.3	0.132	0.108	0.088	14	1	147124	0.004	0.001
3.4	0.088	0.076	0.068	13	0	148892	0.001	0.001
4.1	0.116	0.104	0.088	19	18	136789	0.001	0.001
4.2	0.204	0.168	0.148	20	17	144338	0.001	0.001
4.3	0.180	0.148	0.132	22	7	138070	0.001	0.001
4.4	0.132	0.112	0.096	17	24	135256	0.001	0.001

Example 6. 5000 points with coordinates $(\text{sign}(x) \cdot \sqrt{|x|}, \text{sign}(y) \cdot \sqrt{|y|})$, where x and y are randomly generated integers satisfying $-200 \leq x, y \leq 200$ and $x^2 + y^2 \leq 200^2$.

Example 7. 5000 points with coordinates $(\sqrt{x}, \sqrt{y})$, where x and y are randomly generated integers satisfying $0 \leq x, y \leq 400$, $x^2 + y^2 \leq 400^2$, and $y^2 \leq 3x^2$.

Example 8. The origin $(0, 0)$ and 4999 points with coordinates $(\sqrt{x}, \sqrt{y})$, where x and y are randomly generated integers satisfying $1 \leq x, y \leq 6000$, and $\frac{9}{10} \leq \frac{x}{y} \leq 1$.

For example classes 5 to 8, it was not obvious anymore how to do verification of zero rewriting, since in contrast to computer algebra systems we did not have exact arithmetic in the strong sense for such real algebraic numbers at hand. We use an exact decision number type for real algebraic numbers, namely `leda::real`, which wraps lazy adaptive evaluation with expression dags. Results of experiments for examples 5 to 8 are shown in Table 2.

While Shirayanagi and Sekigawa observe a huge difference in running time between their MAPLE-based versions with and without symbols list, running times of all our `C++`-based versions are roughly on the same order of magnitude. The gain of the variant without symbol lists is evident but much smaller.

Table 2. Convex hull with radical coordinates for four random data sets in example classes 5 to 8. Interval method with correct zero rewriting is based on `leda_bigfloat_intervals` with verification via `leda::real`. Lazy adaptive evaluation is `leda::real`. Running times are for the last successful iteration only.

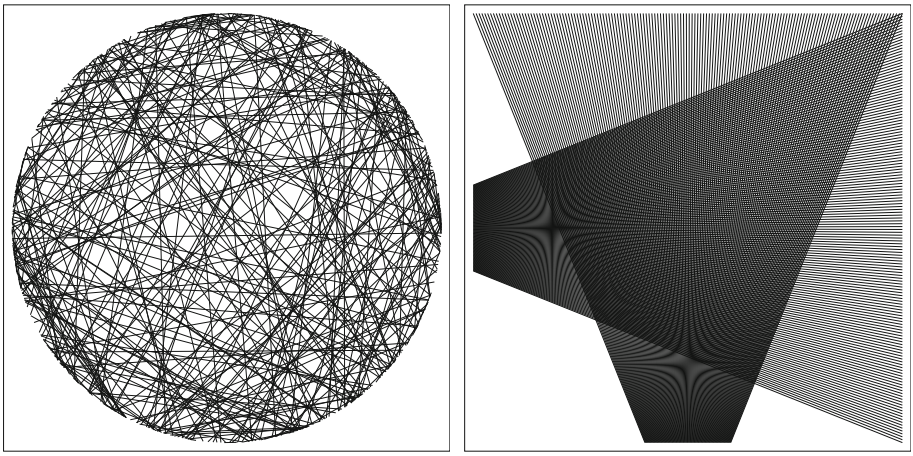
	ISCZ	ISCZ lazy	Without symbol list	Sufficient precision	# zero rewritings	Length of symbol list	Lazy adaptive expr.-dags
5.1	0.656	0.568	0.368	18	19109	241019	0.112
5.2	2.608	2.420	1.296	23	19712	243501	0.112
5.3	2.976	2.768	1.464	24	19952	247434	0.116
5.4	0.604	0.560	0.388	19	21037	221522	0.116
6.1	1.820	1.660	1.024	21	9065	207104	0.080
6.2	0.580	0.520	0.440	19	9064	205591	0.080
6.3	1.468	1.348	0.868	21	8635	203722	0.080
6.4	0.876	0.808	0.576	20	8760	202851	0.080
7.1	0.660	0.612	0.424	18	5167	164377	0.044
7.2	0.556	0.508	0.372	18	4178	161568	0.040
7.3	0.356	0.336	0.288	20	3763	160906	0.036
7.4	0.496	0.456	0.356	20	4351	163229	0.040
8.1	0.672	0.588	0.536	26	52	137443	0.016
8.2	0.396	0.360	0.320	22	48	136970	0.016
8.3	0.472	0.420	0.376	23	48	142530	0.012
8.4	0.476	0.424	0.436	23	55	137351	0.016

Planar convex hull computation is a selective geometric problem of very low computational depth. For such problems floating-point filters are very effective for random input data. Moreover, efficient methods based on error-free transformation techniques and others are known for exact geometric computing of planar convex hulls. These methods are much more efficient than arbitrary precision integer arithmetic and hence much more efficient than (our implementation of) interval-symbol methods with correct zero rewriting. Therefore we consider cascaded geometric computations as well. Such cascaded computations are numerically more demanding and since we do not have access to exact input data anymore in the geometric predicates of later stages, many of the techniques applicable to planar convex hull computation can not be directly used anymore.

Given a set of line segments we compute the convex hull of their intersection points. So the coordinates of the points whose convex hull we are interested in are not part of the input, but numerical values computed during computation. In cascaded computations, we have to record computation history somehow in order to enable verification of zero rewriting, thus in contrast to the previous examples symbol lists are not dispensable anymore. We use CGAL's geometric object generators to create input segments.

Example 9. 300 segments whose endpoints are random points with `double` coordinates (almost) on a circle of radius 250, cf. Fig. 2(a).

Example 10. 150 pairwise disjoint segments with endpoints on vertical line segments with integral x-coordinate and `double` y-coordinates and 150 disjoint segments with endpoints on two horizontal lines with integral y-coordinates and `double` x-coordinates, cf. Fig. 2(b).



(a) (b)

Fig. 2. Segments generated using CGAL's sample code for geometric object generators. (a) 300 segments with endpoints with double coordinates (almost) on circle of radius 250 centered at the origin. (b) 300 segments with endpoints on two vertical and two horizontal segments.

Table 3. Results for examples 9 and 10: computing convex hull of intersection points of segments as shown in Fig. 2.

	ISCZ	Sufficient precision	# zero rewritings	Length of symbol list	Exact rational	Lazy adaptive expr.-dags
9	1.78	31	1283	1748414	2.59	0.31
10	8.97	61	748	2224428	0.56	0.38

Table 3 shows results for examples 9 and 10. In example 9, precision 31 was sufficient, 1283 zero rewritings took place and the running time of the iteration with precision 31 was 1.78 s. Exact computation with `leda::rational` took 2.59 s, which means that interval-symbol method with zero rewriting indeed can save computation time with respect to computation with exact arithmetic if we start at a precision close to sufficient. However, using `leda::real`, running time was 0.31 s only. In example 10, the computation of the convex hull of the intersection points between a vertical fan of segments and a horizontal fan of

segments, we get many collinear points on the 4 convex hull edges. In this example, precision 61 suffices, 748 zero rewritings occurred, and running time was 8.97 s. However, both `leda::rational` and `leda::real` are significantly faster.

We close with a remark on the number of zero rewritings. While there are no zero rewritings with data sets in examples 1 to 3 there are many zero rewritings in examples 5 to 7. Note that we count zero rewritings in the last iteration only, so all these zero rewritings are correct. In all these data sets the number of points we generate is larger than the number of different integer coordinates we allow. So points are not in general position. The resulting degeneracies cause correct zero rewritings. These are the zero rewritings showing up in examples 5 to 7. They do not show up in examples 1 to 3, because the precision of the bigfloat arithmetic suffices to perform integer arithmetic exactly, i.e., we get singleton intervals, especially zero intervals, and there is no need for zero rewriting. Thus, in examples 1 to 3, the bigfloat interval arithmetic already verifies degeneracies, whereas in examples 5 to 7, we have inaccurate approximations only due to the square root operations. Thus, in examples 5 to 7, interval arithmetic does not deliver zero intervals for the point coordinates, and hence coordinate degeneracies cause zero rewritings.

At http://wwwisg.cs.uni-magdeburg.de/ag/ISCZECG the code we use in our experiments is made available. It is based on CGAL 4.10 and LEDA 6.5.

5 Conclusions

Without verification of zero rewriting, interval methods suffer from the same problems as epsilon tweaking, since many geometric algorithms are not robust and inconsistencies can still arise. After all, this non-robustness of geometric algorithms is the motivation for the exact geometric computation paradigm.

With correct zero rewriting interval methods somewhat work like floating-point filters with bigfloat arithmetic. If a zero-containing interval is detected we consult exact computation. However, floating-point filters are lazy. Exact verification of the sign of a value takes place only if the sign is requested, not immediately upon creation. More important, while an incorrect zero rewriting causes a restart from scratch with increased bigfloat precision, a floating-point filter failure just triggers an exact sign computation and overall computation continues based on a decision with the exact sign. There is no restart of the overall algorithm.

Compared to lazy adaptive evaluation with expression dags, interval-symbol methods with correct zero rewriting has a severe performance handicap. With present interval methods with correct zero rewriting all computations are performed with the same precision, the maximum of the minimum precisions required to separate from zero where the maximum is taken over all numerical values arising during computation. With lazy adaptive evaluation we use the precision required for the local sign computation only. This precision is in principle independent of precisions required elsewhere. Degeneracies and near-degeneracies often require higher precision than configurations in general

position. With interval methods with correct zero rewriting such demanding degeneracies and near-degeneracies determine the precision used for all interval computations, including those for less-demanding general position configurations. Lazy adaptive evaluation however always adapts the precision to the situation under investigation and thus uses less precision for general position configurations whenever possible. Furthermore, recording computation history in symbol lists suffers from list blow-up. Expression-dag based exact geometric computation uses reference counting to detect when a numerical value is not used any longer and frees corresponding memory space. Our implementations show that interval-symbol method with correct zero rewriting is manageable in `C++` as well. However, in view of the performance issues discussed above the current approach is most likely not competitive for exact geometric computing, at least for exact geometric computing with algebraic numbers of small algebraic degree.

References

1. Benouamer, M.O., Jaillon, P., Michelucci, D., Moreau, J.M.: A lazy exact arithmetic. In: Proceedings of the 11th Symposium on Computer Arithmetic, pp. 242–249. IEEE (1993)
2. Boost `C++` Libraries. http://www.boost.org/
3. Burnikel, C., Fleischer, R., Mehlhorn, K., Schirra, S.: Efficient exact geometric computation made easy. In: Proceedings of the 15th Symposium on Computational Geometry, pp. 341–350. ACM (1999)
4. Burnikel, C., Funke, S., Mehlhorn, K., Schirra, S., Schmitt, S.: A separation bound for real algebraic expressions. Algorithmica **55**(1), 14–28 (2009)
5. CGAL: Computational Geometry Algorithms Library. http://www.cgal.org/
6. Emiris, I.Z., Mourrain, B., Tsigaridas, E.P.: The DMM bound: multivariate (aggregate) separation bounds. In: Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation, ISSAC 2010, pp. 243–250. ACM, New York (2010). http://doi.acm.org/10.1145/1837934.1837981
7. Fortune, S., van Wyk, C.J.: Static analysis yields efficient exact integer arithmetic for computational geometry. ACM Trans. Graph. **15**(3), 223–248 (1996)
8. Funke, S., Mehlhorn, K.: LOOK: a lazy object-oriented kernel design for geometric computation. Comput. Geom. Theor. Appl. **22**(1–3), 99–118 (2002)
9. Funke, S., Mehlhorn, K., Näher, S.: Structural filtering: a paradigm for efficient and exact geometric programs. Comput. Geom. Theor. Appl. **31**(3), 179–194 (2005)
10. Halperin, D.: Controlled perturbation for certified geometric computing with fixed-precision arithmetic. In: Fukuda, K., Hoeven, J., Joswig, M., Takayama, N. (eds.) ICMS 2010. LNCS, vol. 6327, pp. 92–95. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15582-6_19
11. Halperin, D., Leiserowitz, E.: Controlled perturbation for arrangements of circles. In: Proceedings of the 19th Symposium on Computational Geometry, pp. 264–273. ACM (2003)
12. Halperin, D., Shelton, C.R.: A perturbation scheme for spherical arrangements with application to molecular modeling. Comput. Geom. Theor. Appl. **10**, 273–287 (1998)

13. Karamcheti, V., Li, C., Pechtchanski, I., Yap, C.K.: A core library for robust numeric and geometric computation. In: Proceedings of the 15th Symposium on Computational Geometry, pp. 351–359. ACM (1999)
14. Katayama, A., Shirayanagi, K.: A new idea on the interval-symbol method with correct zero rewriting for reducing exact computations. ACM Commun. Comput. Algebra **50**(4), 176–178 (2016). http://doi.acm.org/10.1145/3055282.3055295
15. Kettner, L., Welzl, E.: One sided error predicates in geometric computing. In: Proceedings of the 15th IFIP World Computer Congress, Fundamentals - Foundations of Computer Science, pp. 13–26 (1998)
16. LEDA: Library of Efficient Data Types and Algorithms. http://www.algorithmic-solutions.com/
17. Li, C., Pion, S., Yap, C.K.: Recent progress in exact geometric computation. J. Logic Algebr. Program. **64**(1), 85–111 (2005)
18. Mehlhorn, K., Osbild, R., Sagraloff, M.: A general approach to the analysis of controlled perturbation algorithms. Comput. Geom. **44**(9), 507–528 (2011). http://www.sciencedirect.com/science/article/pii/S0925772111000460
19. Mignotte, M.: Identification of algebraic numbers. J. Algorithms **3**, 197–204 (1982)
20. Mörig, M.: Algorithm engineering for expression dag based number types. Ph.D. thesis, Otto-von-Guericke-Universität Magdeburg (2015)
21. MPFR: A multiple precision floating-point library. http://www.mpfr.org/
22. Mulmuley, K.: Computational Geometry - An Introduction Through Randomized Algorithms. Prentice Hall, Englewood Cliffs (1994)
23. Pion, S., Fabri, A.: A generic lazy evaluation scheme for exact geometric computations. Sci. Comput. Program. **76**(4), 307–323 (2011)
24. Pion, S., Yap, C.K.: Constructive root bound for k-ary rational input numbers. Theor. Comput. Sci. **369**(1–3), 361–376 (2006)
25. Schirra, S.: Much ado about zero. In: Albers, S., Alt, H., Näher, S. (eds.) Efficient Algorithms. LNCS, vol. 5760, pp. 408–421. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03456-5_27
26. Sekigawa, H.: Zero determination of algebraic numbers using approximate computation and its application to algorithms in computer algebra. Ph.D. thesis, University of Tokyo (2004)
27. Shewchuk, J.R.: Adaptive precision floating-point arithmetic and fast robust geometric predicates. Discrete Comput. Geom. **18**(3), 305–363 (1997)
28. Shirayanagi, K., Sweedler, M.: A theory of stabilizing algebraic algorithms. Technical report, Mathematical Sciences Institute, Cornell University (1995)
29. Shirayanagi, K., Sweedler, M.: Remarks on automatic algorithm stabilization. J. Symb. Comput. **26**(6), 761–765 (1998)
30. Shirayanagi, K.: Floating point Gröbner bases. Math. Comput. Simul. **42**(4–6), 509–528 (1996). https://doi.org/10.1016/S0378-4754(96)00027-4
31. Shirayanagi, K., Sekigawa, H.: A new method of reducing exact computations to obtain exact results. ACM Commun. Comput. Algebra **43**(3/4), 102–104 (2009). http://doi.acm.org/10.1145/1823931.1823950
32. Shirayanagi, K., Sekigawa, H.: Reducing exact computations to obtain exact results based on stabilization techniques. In: Proceedings of Numeric Computation 2009, pp. 191–198 (2009). http://doi.acm.org/10.1145/1577190.1577219
33. Shirayanagi, K., Sekigawa, H.: Interval-symbol method with correct zero rewriting: reducing exact computations to obtain exact results. In: Proceedings of the 18th Asian Technology Conference in Mathematics, pp. 226–235 (2013)

34. Sugihara, K.: Computational geometry in the human brain. In: Akiyama, J., Ito, H., Sakai, T. (eds.) JCDCGG 2013. LNCS, vol. 8845, pp. 145–160. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13287-7_13
35. Sugihara, K., Iri, M.: A robust topology-oriented incremental algorithm for Voronoi diagrams. Int. J. Comput. Geom. Appl. **4**(2), 179–228 (1994)
36. Sugihara, K., Iri, M., Inagaki, H., Imai, T.: Topology-oriented implementation - an approach to robust geometric algorithms. Algorithmica **27**(1), 5–20 (2000)
37. Yap, C.K.: Towards exact geometric computation. Comput. Geom. Theor. Appl. **7**(1–2), 3–23 (1997)
38. Yap, C.K.: Robust geometric computation. In: Goodman, J.E., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, Chap. 41, 2nd edn., pp. 927–952. CRC, Boca Raton (2004)

Sparse Rational Function Interpolation with Finitely Many Values for the Coefficients

Qiao-Long Huang and Xiao-Shan Gao(✉)

KLMM, UCAS, Academy of Mathematics and Systems Science,
Chinese Academy of Sciences, Beijing 100190, China
xgao@mmrc.iss.ac.cn

Abstract. In this paper, we give new sparse interpolation algorithms for black box univariate and multivariate rational functions $h = f/g$ whose coefficients are integers with an upper bound. The main idea is as follows: choose a proper integer β and let $h(\beta) = a/b$ with $\gcd(a, b) = 1$. Then f and g can be computed by solving the polynomial interpolation problems $f(\beta) = ka$ and $g(\beta) = ka$ for some unique integer k. Experimental results show that the univariate interpolation algorithm is almost optimal.

1 Introduction

The interpolation for a sparse multivariate rational function $h = f/g$ given as a black box is a basic computational problem [1,2,6–8]. Here, sparse means that an upper bound for the number of terms in f and g is given. In many interpolation algorithms, an upper bound for the degrees of f and g is also given.

In [3], a new constraint in sparse interpolation is considered: it is assumed that the coefficients of the sparse polynomial are taken from a known finite set. The method in [3] can be considered as a generalization for Kronecker's idea of interpolating polynomials [9]. Comparing to [9], the main contributions of [3] are that polynomials with rational numbers as coefficients could be interpolated and an improved Kronecker type substitution is used to reduce multivariate interpolation to univariate interpolation.

In this paper, we extend the algorithm in [3] to rational functions. Consider the interpolation of $h = f/g \in \mathbb{Z}(x_1, x_2, \ldots, x_n)$, where T, D, C are upper bounds for the terms, degrees, and the absolute values of the coefficients of f and g, respectively. The main idea of the algorithm is reduce the interpolation of h into that of f and g.

In the univariate case, let $\beta \in \mathbb{N}^+$, $h(\beta) = a/b, \gcd(a, b) = 1$, and $\mu = \gcd(f(\beta), g(\beta))$. We prove that if $\beta \geq 2TC^2 + 1$, then for $k \in \mathbb{N}\,(k \leq \mu)$, $k = \mu$ if and only if there exist $p, q \in \mathbb{Z}[x]$ such that $p(\beta) = ka$, $q(\beta) = kb$, and the coefficients of p and q are bounded by C. Thus we can find μ by computing univariate polynomials $p(\beta) = ka$, $q(\beta) = kb$ for $k = 1, 2, \ldots$ and check whether the coefficients of p and q are bounded by C. The value for β can be further

Partially supported by a grant from NSFC No. 11688101.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 227–242, 2017.
https://doi.org/10.1007/978-3-319-72453-9_16

reduced in two ways. If we evaluate h at two sample points $h(\beta)$ and $h(\beta+1)$, then β can be taken as $\beta = \lceil\sqrt{2T}C\rceil$. For $\beta = 3C+1$, we can obtain a probabilistic algorithm.

In the multivariate case, similar idea is used to give a probabilistic algorithm. The sample point used is $\beta_i = (\beta + c_i)^{(2D+1)^{i-1}}, i = 1, 2, \ldots, n$, where $\beta = 2TC^2+1$, and $c_1 \leq c_2 \leq \cdots \leq c_n$ are random numbers. We show that with high probability, we can recover h from $h(\beta_1, \ldots, \beta_n)$. The substitution is a variant of Kronecker's substitution [9], where the shifts $\beta + c_i$ are introduced to avoid the appearance of common factors for the numerator and numerator after the substitution. New Kronecker type substitutions can also be found in [5].

The arithmetic complexity of the univariate interpolation is $\mathcal{O}(\mu T \log^2 D)$ and the length of the data is $\mathcal{O}(D(\log C + \log T))$. The arithmetic complexity of the multivariate interpolation is $\mathcal{O}(\mu n T \log^2 D)$ and the length of the data is $\mathcal{O}(D^n \log(TC^2 + N)$.

Extensive experiments are done for the algorithms. It is shown that the univariate interpolation algorithm is almost optimal in the sense that the time for interpolating f/g is almost the same as that of interpolating f and g, which means that μ is small in most cases. When μ is small, the bit complexity is linear in TD, which is optimal. In the multivariate case, the algorithm is less sensitive for T but is quite sensitive for D and n, because the sample data is of height D^n due to the use of the Kronecker substitution.

2 Preliminary Algorithms

In this section, we will present some preliminary algorithms which will be used in this paper. Assume $f(x) = c_1 x^{d_1} + c_2 x^{d_2} + \cdots + c_t x^{d_t}$, where $d_1, d_2, \ldots, d_t \in \mathbb{N}, d_1 < d_2 < \cdots < d_t$, and $c_1, c_2, \cdots, c_t \in A$, where $A \subset \mathbb{C}$ is a finite set. Introduce the following notations

$$C := \max_{a \in A}(|a|), \quad \varepsilon := \min(\varepsilon_1, \varepsilon_2) \tag{1}$$

where $\varepsilon_1 := \min_{a,b \in A, a \neq b} |a - b|$ and $\varepsilon_2 := \min_{a \in A, a \neq 0} |a|$. We have

Theorem 1 ([3]). *If $\beta \geq \frac{2C}{\varepsilon} + 1$, then $f(x)$ can be uniquely determined by $f(\beta)$.*

Based on the above theorem, the following interpolation algorithm for polynomials in $\mathbb{Q}[x]$ is given in [3], which is needed in this paper.

Algorithm 2 (UPolySIRat) *[3, Algorithm 2.14]*

Input: $H, C \in \mathbb{N}$, $\beta = 2CH(H-1)$, and $\rho = f(\beta)$ for a black box polynomial $f(x) \in \mathbb{Q}[x]$ whose coefficients are in $A = \{\frac{b}{a} \,|\, 0 < a \leq H, |\frac{b}{a}| \leq C, a, b \in \mathbb{Z}\}$.
Output: The exact form of $f(x)$.

Theorem 3 ([3]). *The arithmetic complexity of Algorithm 2 is $\mathcal{O}(t \log^2 d \log H)$ and the bit complexity is $\widetilde{\mathcal{O}}(td \log H(\log C + \log H))$, where $d = \deg(f)$ and $t = \#f$.*

We can compute the degree of $f(x)$ as follows.

Lemma 1 ([3]). *Assume* $\beta \geq \frac{2C}{\varepsilon} + 1$. *If* $k \leq d_t$, *then* $|\frac{f(\beta)}{\beta^k}| > \frac{\varepsilon}{2}$; *if* $k > d_t$, *then* $|\frac{f(\beta)}{\beta^k}| < \frac{\varepsilon}{2}$. *In particular,* $d_t = \lfloor \log_\beta 2|f(\beta)| \rfloor$.

We need the following special case of Algorithm 2, where $f \in \mathbb{Z}[x]$.

Algorithm 4 (UPolySIInt)

Input: $\beta \geq 2C + 1$, $\rho = f(\beta) \in \mathbb{Z}$, a variable x, an upper bound $C \geq \|f\|_\infty$.
Output: The exact form of $f(x)$, or failure if the polynomial f obtained satisfying $\|f\|_\infty > C$.

The following theorem is a corollary of Theorem 3.

Theorem 5. *The arithmetic complexity of Algorithm 4 is* $\mathcal{O}(t \log^2 d)$ *and the bit complexity is* $\widetilde{\mathcal{O}}(td \log C)$, *where* $t = \#f$, $d = \deg(f)$.

3 Univariate Rational Function Interpolation

In this section, we give several sparse interpolation algorithms for univariate rational functions.

3.1 A Basic Interpolation Algorithm

In this subsection, we give a polynomial-time deterministic interpolation algorithm which is the starting point for more efficient algorithms.

We first introduce some notations. In this paper, for $f(x), g(x) \in \mathbb{Z}[x]$, $\gcd(f, g)$ also contains the greatest common factor of the coefficients of f and g. Let $h = \frac{f(x)}{g(x)} \in \mathbb{Z}(x)$ be a rational function, where $\gcd(f, g) = 1$. Denote $\deg(h) := \max\{\deg(f), \deg(g)\}$, $\#h := \max\{\#f, \#g\}$, $\|h\|_\infty := \max\{\|f\|_\infty, \|g\|_\infty\}$, where $\#f$ is the number of the terms of f and $\|f\|_\infty$ is the maximal absolute value of the coefficients of f.

For a positive integer β, let $h(\beta) = \frac{a}{b}$, where $a, b \in \mathbb{Z}$ and $\gcd(a, b) = 1$. Let $\mu = \gcd(f(\beta), g(\beta)) > 0$. Then, we have

$$a = \frac{f(\beta)}{\mu}, b = \frac{g(\beta)}{\mu} \tag{2}$$

Denote $f_1(x) := \frac{1}{\mu} f(x), g_1(x) := \frac{1}{\mu} g(x)$. If $\|h\|_\infty \leq C$, then $f_1(\beta) = a, g_1(\beta) = b$, and the coefficients of $f_1(x), g_1(x)$ are in $\{\frac{a}{\mu} \in \mathbb{Z} || a| \leq C\}$. If we can give an upper bound H for μ and let $\beta \geq 2CH(H-1)+1$, then we can recover f_1 and g_1 using the Algorithm 2 and hence $f/g = f_1/g_1$. Therefore, a key issue in sparse interpolation for rational functions is to determine an upper bound for μ. The following lemmas give such an estimation.

Lemma 2 [10, p. 147]. *Let* $f, g \in \mathbb{Z}[x]$, *and* $n = \deg(f) \geq m = \deg(g) \geq 1$. *Then* $|\mathbf{res}(f, g, x)| \leq (n+1)^{m/2}(m+1)^{n/2} \parallel f \parallel_\infty^m \parallel g \parallel_\infty^n$.

Lemma 3. *If* $f, g \in \mathbb{Z}[x]$, *and* $D \geq \max\{\deg(f), \deg(g)\}$, $C \geq \max\{\|f\|_\infty, \|g\|_\infty\}$, *then* $\mu \leq (D+1)^D C^{2D}$, *where* μ *is defined in (2).*

Proof. Since $\gcd(f,g) = 1$, $\mathbf{res}(f,g,x) \neq 0$. By [10, p.147], there exist two nonzero polynomials $s, t \in \mathbb{Z}[x]$, such that $fs + gt = \mathbf{res}(f,g,x)$. So we have $f(\beta)s(\beta) + g(\beta)t(\beta) = \mathbf{res}(f,g,x)$. Since $\mathbf{res}(f,g,x)$ is an integer, we have $\gcd(f(\beta), g(\beta))|\mathbf{res}(f,g,x)$. By Lemma 2, $\mu \leq |\mathbf{res}(f,g,x)| \leq (D+1)^D C^{2D}$. □

Theorem 6. *Let* $h(x) \in \mathbb{Z}(x)$ *with* $D \geq \deg(h)$ *and* $C \geq \|h\|_\infty$. *Denote* $H := (D+1)^D C^{2D}$. *If* $\beta \geq 2CH(H-1)+1$, *then* $h(x)$ *can be recovered from* $h(\beta)$.

Proof. Use the notations in (2). If we can interpolate the polynomials $\frac{1}{\mu}f(x), \frac{1}{\mu}g(x)$ from the values a and b, then we finish the interpolation. By Lemma 3, we know $|\mu| \leq H$, so the coefficients of $\frac{1}{\mu}f, \frac{1}{\mu}g$ are in the finite set $\{\frac{s}{t} \in \mathbb{Q} || t| \leq H, |s| \leq C\}$. Let $\varepsilon = \frac{1}{H(H-1)}$, when $\beta \geq 2CH(H-1)+1$, we can interpolate $\frac{1}{\mu}f(x), \frac{1}{\mu}g(x)$ from a, b with Algorithm 2. Thus, $h(x)$ can be recovered from $h(\beta)$. □

We now give the algorithm.

Algorithm 7 (URFunSI0)

Input: A black box $h \in \mathbb{Z}(x)$, $D, C \in \mathbb{N}$, where $D \geq \deg(h), C \geq \|h\|_\infty$.
Output: The exact form of $h(x)$.

Step 1: Let $H := (D+1)^D C^{2D}, \beta := 2CH(H-1)+1$. Evaluate $h(\beta) = \frac{a}{b}$.
Step 2: Let $f(x) := \mathbf{UPolySIRat}(C, H, \beta, a), g(x) := \mathbf{UPolySIRat}(C, H, \beta, b)$.
Step 3: Return $\frac{f(x)}{g(x)}$.

Theorem 8. *The arithmetic complexity of Algorithm 7 is* $\mathcal{O}(TD\log C + TD\log D)$ *and the bit complexity is* $\widetilde{\mathcal{O}}(TD^3\log^2 C)$.

Proof. By Theorem 3, the arithmetic complexity of Algorithm **UPolySIRat** is $\mathcal{O}(T\log H)$ and bit complexity is $\mathcal{O}(TD(\log H \log C + \log^2 H))$. Since $H = (D+1)^D C^{2D}$, the theorem follows immediately. □

It should be pointed out that Theorem 8 is a theoretical result, since the number β is too large. Practical algorithms will be given in the following sections, which are modifications of Algorithm 7.

3.2 Deterministic Incremental Interpolation

In Algorithm 7, we use an upper bound for μ. In this section, an algorithm will be given, where μ will be searched incrementally. We first give a lemma.

Lemma 4. *Let* $f, f_1, g, g_1 \in \mathbb{Z}[x]$, $\mathbf{gcd}(f,g) = 1, \deg(f_1) \leq \deg(f), \deg(g_1) \leq \deg(g)$. *If* $\frac{f_1}{g_1} = \frac{f}{g}$, *then there is a nonzero integer* δ, *such that* $f_1 = \delta f, g_1 = \delta g$.

Proof. Since $\frac{f_1}{g_1} = \frac{f}{g}$, we have $f_1 g = g_1 f$ and hence $f | g f_1$. Since $\gcd(f, g) = 1$, we have $f | f_1$. From $\deg(f_1) \leq \deg(f)$, there exists a rational number $\frac{a}{b}$, such that $f_1 = \frac{a}{b} f$. For the same reason we have $g_1 = \frac{a}{b} g$. Since $f_1, g_1 \in \mathbb{Z}[x]$, all their coefficients are integers. So b divides all the coefficients of f, g, as $\gcd(f, g) = 1$, and hence $b = \pm 1$. So $\delta = \frac{a}{b}$ is an integer. □

Theorem 9. *Let $h(x) = \frac{f(x)}{g(x)} \in \mathbb{Z}(x)$, $T \geq \#h, C \geq \|h\|_\infty$. If $\beta \geq 2TC^2 + 1$, then $h(x)$ can be recovered from $h(\beta)$.*

Proof. Let a, b, μ be introduced in (2). We claim that for $i = 1, 2, \cdots$, only when $i = \mu$, the values $a \cdot i, b \cdot i$ correspond to two polynomials with coefficients bounded by C. We prove the claim by contradiction. Assume there exists an $i_0 < \mu$, such that $a \cdot i_0, b \cdot i_0$ corresponding to two polynomials f_1, g_1 in $\mathbb{Z}[x]$ with $C \geq \|f_1\|_\infty, \|g_1\|_\infty$. Since $|i_0 a| < |\mu a|, |i_0 b| < |\mu b|$, we have $\deg(f_1) \leq \deg(f), \deg(g_1) \leq \deg(g)$ by Lemma 1. Then we have $\frac{f(\beta)}{g(\beta)} = \frac{f_1(\beta)}{g_1(\beta)}$. This can be changed into $f(\beta) g_1(\beta) = f_1(\beta) g(\beta)$. If we let $w(x) := f(x) g_1(x)$, $v(x) := f_1(x) g(x)$, then $w(\beta) = v(\beta)$. Since $T \geq \#h$, $TC^2 \geq \|w\|_\infty, \|v\|_\infty$. Since $\beta \geq 2TC^2 + 1$, we have $w(x) = v(x)$, which can be changed into $\frac{f(x)}{g(x)} = \frac{f_1(x)}{g_1(x)}$. By the Lemma 4, we have $f_1(x) = \delta f(x), g_1(x) = \delta g(x)$, where δ is a nonzero integer, then $|i_0 a| = |f_1(\beta)| = |\delta f(\beta)| \geq |f(\beta)| = |\mu a|$. This is a contradiction, so we prove the theorem. □

Theorem 9 leads to the following deterministic algorithm.

Algorithm 10 (URFunSI1)

Input: A black box $h(x) \in \mathbb{Z}(x)$, $T, C \in \mathbb{N}$, where $T \geq \#h, C \geq \|h\|_\infty$.
Output: The exact form of $h(x)$.

Step 1: Let $\beta := 2TC^2 + 1$.
Step 2: Evaluate $h(\beta)$, assume $h(\beta) = \frac{a}{b}$.
Step 3: Let $i = 1$;
Step 4: $f :=$ **UPolySInt**$(\beta, a \cdot i, x, C)$;
  if ($f = failure$ **or** $\#f > T$) **then** $i := i + 1$; go to Step 4; **end if**
Step 5: $g :=$ **UPolySIInt**$(\beta, b \cdot i, x, C)$;
  if ($g = failure$ **or** $\#g > T$) **then** $i := i + 1$; go to Step 4; **end if**
Step 6: **return** $\frac{f}{g}$.

Theorem 11. *The arithmetic complexity of Algorithm 10 is $\mathcal{O}(\mu T \log^2 D)$, and the height of the data is $\mathcal{O}(D(\log C + \log T))$, where μ is defined in (2). In particular, when $\mu = 1$, the arithmetic complexity is $\mathcal{O}(T \log^2 D)$.*

Proof. Since it calls μ Algorithm 4, by Theorem 5, the arithmetic complexity is $\mathcal{O}(\mu T \log^2 D)$. Since $\beta = 2TC^2 + 1$, $f(\beta)$ is $\mathcal{O}(C(TC^2)^D)$ and the height of the data is $\mathcal{O}(D(\log C + \log T))$. □

3.3 Deterministic Incremental Interpolation with Two Points

In Algorithm 10, we recover $h(x)$ from $h(\beta)$ for $\beta = 2TC^2 + 1$. In this section, we show that $h(x)$ can be recovered from $h(\beta)$ and $h(\beta+1)$ for a much smaller $\beta = \lceil\sqrt{2T}C\rceil$. The following lemma shows how to recover a polynomial from two smaller points.

Lemma 5. *Let* $f(x) = \sum_{i=1}^{t} c_i x^{d_i} \in \mathbb{Z}[x], d_1 < d_2 < \cdots < d_t$, *and* $C \geq \|f\|_\infty$. *If* $\beta \geq \sqrt{2C}$, *then* $f(x)$ *can be recovered from* $f(\beta)$ *and* $f(\beta+1)$.

Proof. Assume that there exists another $g(x) = \sum_{i=1}^{s} a_i x^{k_i} \in \mathbb{Z}[x], k_1 < k_2 < \cdots < k_s$, and $C \geq \|g\|_\infty$, such that $g(\beta) = f(\beta), g(\beta+1) = f(\beta+1)$. Firstly, we prove $d_1 = k_1$. It is clear that d_1 (k_1) is the largest integer such that $\mathbf{mod}(f(\beta), \beta^{d_1}) = \mathbf{mod}(f(\beta+1), (\beta+1)^{d_1}) = 0$ ($\mathbf{mod}(g(\beta), \beta^{k_1}) = 0, \mathbf{mod}(g(\beta+1), (\beta+1)^{k_1}) = 0$). Since $f(\beta) = g(\beta), f(\beta+1) = g(\beta+1)$, we have $d_1 = k_1$. Next, we prove $a_1 = c_1$. From $\frac{f(\beta)}{\beta^{d_1}}$ **mod** $\beta = c_1$, $\frac{g(\beta)}{\beta^{d_1}}$ **mod** $\beta = a_1$, $\frac{f(\beta+1)}{(\beta+1)^{d_1}}$ **mod** $(\beta+1) = c_1$, $\frac{g(\beta+1)}{(\beta+1)^{d_1}}$ **mod** $(\beta+1) = a_1$, we have $(a_1 - c_1)$ **mod** $\beta = 0$ and $(a_1 - c_1)$ **mod** $(\beta+1) = 0$. Since $\gcd(\beta, \beta+1) = 1$, we have $(a_1 - c_1)$ **mod** $\beta(\beta+1) = 0$. $|a_1|, |c_1| \leq C$, so $|a_1 - c_1| \leq 2C$. But $|\beta(\beta+1)| \geq \sqrt{2C}(\sqrt{2C}+1) > 2C$, so $a_1 = c_1$. The other terms can be proved by induction. □

Theorem 12. *Let* $h(x) = \frac{f(x)}{g(x)} \in \mathbb{Z}(x)$, $T \geq \#h, C \geq \|h\|_\infty$. *If* $\beta \geq \lceil\sqrt{2T}C\rceil$, $h(x)$ *can be recovered from* $h(\beta)$ *and* $h(\beta+1)$.

Proof. Use the same notations as Theorem 9. We still prove it by contradiction. Assume there exists an $i_0 < \mu$, such that $a \cdot i_0, b \cdot i_0$ correspond to two integer polynomials with $C \geq \|f_1\|_\infty, \|g_1\|_\infty$. Since $|i_0 a| < |\mu a|, |i_0 b| < |\mu b|$, we have $\deg(f_1) \leq \deg(f), \deg(g_1) \leq \deg(g)$. Then we have $\frac{f(\beta)}{g(\beta)} = \frac{f_1(\beta)}{g_1(\beta)}, \frac{f(\beta+1)}{g(\beta+1)} = \frac{f_1(\beta+1)}{g_1(\beta+1)}$. This can be change to $f(\beta)g_1(\beta) = f_1(\beta)g(\beta), f(\beta+1)g_1(\beta+1) = f_1(\beta+1)g(\beta+1)$. Let $w(x) := f(x)g_1(x)$, $v(x) := f_1(x)g(x)$. Then $w(\beta) = v(\beta), w(\beta+1) = v(\beta+1)$. Since $T \geq \max\{\#f, \#g\}$, $TC^2 \geq \max\{\|w\|_\infty, \|v\|_\infty\}$. From $\beta \geq \lceil\sqrt{2T}C\rceil$, by Lemma 5, we have $w(x) = v(x)$, or $\frac{f(x)}{g(x)} = \frac{f_1(x)}{g_1(x)}$. By Lemma 4, the same reason as Theorem 9, we prove the theorem. □

Based on the above theorem, an interpolation algorithm using two points can be given. In the following algorithm, we assume $T \geq 5$. In this case, $\sqrt{2T}C \geq 2C+1$, so the evaluation satisfies the input condition of Algorithm **UPolySIInt**.

Algorithm 13 (URFunSI2)

Input: A black box $h(x) \in \mathbb{Z}(x)$, T, C, where $T \geq \#h, C \geq \|h\|_\infty$.
Output: The exact form of $h(x)$.

Step 1: Let $T_1 := \max(T, 5)$, $\beta := \lceil\sqrt{2T_1}C\rceil$.
Step 2: Evaluate $h(\beta), h(\beta+1)$ and assume $h(\beta) = \frac{a_1}{b_1}, h(\beta+1) = \frac{a_2}{b_2}$.

Step 3: $i = 1$;
Step 4: $f := \mathbf{UPolySIInt}(\beta, a_1 \cdot i, x, C)$;
if ($f = failure$ **or** $\#f > T$) **then** $i := i + 1$; go to Step 4; **end if**
Step 5: $g := \mathbf{UPolySIInt}(\beta, b_1 \cdot i, x, C)$;
if ($g = failure$ **or** $\#g > T$) **then** $i := i + 1$; go to Step 4; **end if**
Step 6: **if** $\frac{f(\beta+1)}{g(p+1)} = \frac{a_2}{b_2}$ **then** return $\frac{f}{g}$; **else** $i := i + 1$; go to Step 4.

Theorem 14. *The arithmetic complexity of Algorithm 13 is* $\mathcal{O}(\mu T \log^2 D)$*, and the length of the data is* $\mathcal{O}(D(\log C + \log T))$*. In particular, when* $\mu = 1$*, the arithmetic complexity is* $\mathcal{O}(T \log^2 D)$*.*

Proof. The analysis of arithmetic complexity is the same as Theorem 11. □

Note that the complexity of Algorithm 13 is the same as that of Algorithm 10, but Algorithm 13 is practically much faster than Algorithm 10 as shown in Sect. 5.

3.4 Probabilistic Univariate Rational Function Interpolation

In Algorithms 10 and 13, $\beta = 2TC^2 + 1$ and $\beta = \lceil \sqrt{2T}C \rceil$, respectively. In this section, we will give a probabilistic algorithm where $\beta = 3C + 1$ under the condition that a degree bound for f is known.

Lemma 6. *Assume* $h(x) = \frac{f(x)}{g(x)} \in \mathbb{Z}(x)$*,* $C \geq \|h\|_\infty, D \geq \deg(f)$*. Let* $\beta \geq 2C + 1$*,* $h(\beta) = \frac{a}{b}$*, and* $\mu = \gcd(f(\beta), g(\beta))$*. Then* $|\mu| \leq \lfloor \frac{\beta^{D+1}}{2|a|} \rfloor$*.*

Proof. By Lemma 1, $|\frac{f(\beta)}{\beta^{D+1}}| < \frac{1}{2}$. Since $h(\beta) = \frac{a}{b}$, we have $a = \frac{1}{\mu} f(\beta)$, and $\frac{|a|}{\beta^{D+1}} = |\frac{\frac{1}{\mu} f(\beta)}{\beta^{D+1}}| < \frac{1}{2|\mu|}$. Then we can give an upper bound $|\mu| < \frac{\beta^{D+1}}{2|a|}$. Since μ is an integer, $|\mu| \leq \lfloor \frac{\beta^{D+1}}{2|a|} \rfloor$. □

We can give a lower bound of degree of $f(x)$. Assume $h(\beta) = \frac{a}{b}$. By Lemma 1, the smallest number d satisfying $\frac{|a|}{\beta^{d+1}} < \frac{1}{2}$ is a lower degree bound of $f(x)$. The lower and upper degree bounds will avoid lots of computing.

In this subsection, we use two points $h(\beta), h(\beta + 1)$ to interpolate $h(x)$. The following theorems will show some relations between the two points.

Lemma 7. *Let* $f(x) = \sum_{i=1}^{t} c_i x^{d_i} \in \mathbb{Z}[x], C \geq \|f\|_\infty, d_1 < d_2 < \cdots < d_t$*. If* $\beta \geq 2C + 1$ *and* $Q := \frac{f(\beta)/\beta^{d_t}}{f(\beta+1)/(\beta+1)^{d_t}}$*,* $E := 1 + \frac{2C}{\beta(\beta-1)}$*, then* $\frac{1}{E} < Q < E$*.*

Proof. Denote $q_1 := \frac{\sum_{i=1}^{t-1} c_i \beta^{d_i}}{\beta^{d_t}}$ and $q_2 := \frac{\sum_{i=1}^{t-1} c_i (\beta+1)^{d_i}}{(\beta+1)^{d_t}}$. Then $Q = \frac{f(\beta)/\beta^{d_t}}{f(\beta+1)/(\beta+1)^{d_t}}$ $= \frac{c_t+q_1}{c_t+q_2} = 1 + \frac{q_1-q_2}{c_t+q_2}$. Since $|c_t| \geq 1, |q_2| < \frac{\varepsilon}{2} = \frac{1}{2}$(by Lemma 1), we have $|c_t + q_2| > \frac{1}{2}$. So $|\frac{q_1-q_2}{c_t+q_2}| < 2|q_1 - q_2|$. From $|q_1 - q_2| = |\sum_{i=1}^{t-1} c_i (\frac{1}{\beta^{d_t-d_i}} - \frac{1}{(\beta+1)^{d_t-d_i}})| \leq C|\sum_{i=1}^{t-1} (\frac{1}{\beta^{d_t-d_i}} - \frac{1}{(\beta+1)^{d_t-d_i}})| \leq C|\sum_{i=1}^{d_t} (\frac{1}{\beta^i} - \frac{1}{(\beta+1)^i})| = C\sum_{i=1}^{d_t} \frac{1}{\beta^i} - C\sum_{i=1}^{d_t}$

$\frac{1}{(\beta+1)^i} = C\frac{\frac{1}{\beta}-\frac{1}{\beta^{d_t+1}}}{1-\frac{1}{\beta}} - C\frac{\frac{1}{\beta+1}-\frac{1}{(\beta+1)^{d_t+1}}}{1-\frac{1}{\beta+1}} = C\frac{1-\frac{1}{\beta^{d_t-1}}+\frac{\beta-1}{(\beta+1)^{d_t}}}{\beta(\beta-1)} < \frac{C}{\beta(\beta-1)}$. So $|Q-1| \le 2|q_1-q_2| < \frac{2C}{\beta(\beta-1)}$, so we prove the first inequality. Note that $\frac{1}{Q} = \frac{f(\beta+1)/(\beta+1)^{d_t}}{f(\beta)/\beta^{d_t}} = \frac{c_t+q_2}{c_t+q_1} = 1+\frac{q_2-q_1}{c_t+q_1}$. We also have $|\frac{q_2-q_1}{c_t+q_1}| < \frac{2C}{\beta(\beta-1)}$. So $|\frac{1}{Q}-1| < \frac{2C}{\beta(\beta-1)}$. Now we have $1-\frac{2C}{\beta(\beta-1)} < Q < 1+\frac{2C}{\beta(\beta-1)}$ and $1-\frac{2C}{\beta(\beta-1)} < \frac{1}{Q} < 1+\frac{2C}{\beta(\beta-1)}$, which is $1-\frac{2C}{\beta(\beta-1)} < Q < 1+\frac{2C}{\beta(\beta-1)}$ and $\frac{1}{1+\frac{2C}{\beta(\beta-1)}} < Q < \frac{1}{1-\frac{2C}{\beta(\beta-1)}}$. Since $1+\frac{2C}{\beta(\beta-1)} \le \frac{1}{1-\frac{2C}{\beta(\beta-1)}}$ and $1-\frac{2C}{\beta(\beta-1)} \le \frac{1}{1+\frac{2C}{\beta(\beta-1)}}$, we proved the lemma. □

Lemma 8. *Let* $h(x) = \frac{f(x)}{g(x)} \in \mathbb{Z}(x)$ *and* $h(\beta) = \frac{a_1}{b_1}, h(\beta+1) = \frac{a_2}{b_2}$, *where* $\gcd(a_1,b_1)=1, \gcd(a_2,b_2)=1$, $D \ge \deg(f) \ge d$. *If* $a_1 = \frac{f(\beta)}{\mu_1}, a_2 = \frac{f(\beta+1)}{\mu_2}$ *and* $Q_1 := \frac{a_1/\beta^d}{a_2/(\beta+1)^d}, Q_2 := \frac{a_1/\beta^D}{a_2/(\beta+1)^D}$, $E := 1+\frac{2C}{\beta(\beta-1)}$, *then* $|Q_1|\frac{1}{E} < |\frac{\mu_2}{\mu_1}| < |Q_2|E$.

Proof. Let $Q := \frac{f(\beta)/\beta^{d_t}}{f(\beta+1)/(\beta+1)^{d_t}}$. Then we have $Q_1 = \frac{\mu_2}{\mu_1}\frac{f(\beta)}{f(\beta+1)}\frac{(\beta+1)^{d_t}}{\beta^{d_t}}\frac{\beta^{d_t-d}}{(\beta+1)^{d_t-d}} = Q\frac{\mu_2}{\mu_1}\frac{\beta^{d_t-d}}{(\beta+1)^{d_t-d}}$ and $Q_2 = \frac{\mu_2}{\mu_1}\frac{f(\beta)}{f(\beta+1)}\frac{(\beta+1)^{d_t}}{\beta^{d_t}}\frac{(\beta+1)^{D-d_t}}{\beta^{D-d_t}} = Q\frac{\mu_2}{\mu_1}\frac{(\beta+1)^{D-d_t}}{\beta^{D-d_t}}$. By Lemma 7, $\frac{1}{E} < Q < E$. Then $|Q_1| < |\frac{\mu_2}{\mu_1}|\frac{\beta^{d_t-d}}{(\beta+1)^{d_t-d}}E \Rightarrow |\frac{\mu_2}{\mu_1}| > |Q_1|\frac{(\beta+1)^{d_t-d}}{\beta^{d_t-d}}\frac{1}{E} \ge |Q_1|\frac{1}{E}$ and $|Q_2| > |\frac{\mu_2}{\mu_1}|\frac{(\beta+1)^{D-d_t}}{\beta^{D-d_t}}\frac{1}{E} \Rightarrow |\frac{\mu_2}{\mu_1}| < |Q_2|\frac{\beta^{D-d_t}}{(\beta+1)^{D-d_t}}E \le |Q_2|E$. □

It is easy to see that we have the best result if $D = d = \deg(f(x))$.

Corollary 1. *If* $|Q_1| \ge E$, *then* $|\mu_2| > |\mu_1|$. *If* $|Q_2| \le \frac{1}{E}$, *then* $|\mu_2| < |\mu_1|$.

Proof. By Lemma 8, we have $|Q_1|\frac{1}{E} < |\frac{\mu_2}{\mu_1}| < |Q_2|E$, and the lemma follows from this. □

Now we give the algorithm.

Algorithm 15 (URFunSIP)

Input: A black box $h(x) = \frac{f(x)}{g(x)} \in \mathbb{Z}(x)$, D, C, where $D \ge \deg(h), C \ge \|h\|_\infty$.
Output: The exact form of $h(x)$ or a wrong rational function.

Step 1: Let $\beta := 3C+1$.
Step 2: Evaluate $h(\beta), h(\beta+1)$, and assume $h(\beta) = \frac{a_1}{b_1}, h(\beta+1) = \frac{a_2}{b_2}$.
Step 3: Let $d = \max(\lfloor\log_\beta(2a_1)\rfloor, \lfloor\log_{\beta+1}(2a_2)\rfloor)$(due to Lemma 1), $k_1 = \lfloor\frac{\beta^{D+1}}{|a_1|}\rfloor$, $k_2 = \lfloor\frac{(\beta+1)^{D+1}}{|a_2|}\rfloor$, $Q_1 = |\frac{a_1/\beta^d}{a_2/(\beta+1)^d}|$, $Q_2 = |\frac{a_1/\beta^D}{a_2/(\beta+1)^D}|$, $E = 1+\frac{2C}{\beta(\beta-1)}$.
Step 4: Let $i := 1$. **If** $Q_1 \ge E$ **then** goto step 5; **If** $Q_2 \le \frac{1}{E}$ **then** goto step 6. **If** $k_1 < k_2$, **then** goto step 5; **Else** goto step 6.
Step 5: **while** an integer in $(\frac{Q_1}{E}i, Q_2Ei)$ **do**
a: Let $f :=$ **UPolySIInt**$(\beta, a_1\cdot i, x, C)$ and $g :=$ **UPolySIInt**$(\beta, b_1\cdot i, x, C)$.
b: **if** $f = failure$ or $g = failure$ **then** $i := i+1$; goto step 5;
c: **if** $h(\beta+1) = \frac{f(\beta+1)}{g(\beta+1)}$, **then** return $\frac{f(x)}{g(x)}$.

Step 6: **while** an integer in $(\frac{1}{Q_2 E} i, \frac{E}{Q_1} i)$ **do**

a: Let $f := \mathbf{UPolySIInt}(\beta+1, a_2 \cdot i, x, C)$ and $g := \mathbf{UPolySIInt}(\beta+1, b_2 \cdot i, x, C)$;

b: **if** $f = failure$ or $g = failure$ **then** $i := i+1$; goto step 6;

c: **if** $h(\beta) = \frac{f(\beta)}{g(\beta)}$ **then** return $\frac{f(x)}{g(x)}$.

Theorem 16. *The algorithm is correct. The arithmetic complexity of the algorithm is* $\mathcal{O}(\mu D \log^2 D)$*, where* $\mu \le (1+\frac{1}{3C})^{D-d+2} \min\{\mu_1, \mu_2\}$*. The height of the data is* $\mathcal{O}(D \log C)$*. In particular, when* $\mu = 1$*, the arithmetic complexity is* $\mathcal{O}(T \log^2 D)$*.*

Proof. Assume $\gcd(a_1, b_1) = 1, f(\beta) = \mu_1 a_1, g(\beta) = \mu_1 b_1$, $\gcd(a_2, b_2) = 1, f(\beta+1) = \mu_2 a_2, g(\beta+1) = \mu_2 b_2$, and $\mu_1, \mu_2 > 0$. The main idea of the algorithm is to find one of μ_1, μ_2, and thus the exact value $f(\beta)$ or $f(\beta+1)$. Since $\beta \ge 2C+1$, we can recover $f(x)$ and $g(x)$ by Algorithm 4. In this algorithm, we use an incremental approach to find the probably smaller one in $\{\mu_1, \mu_2\}$. We give some simple criterions to compare which one is small due to Corollary 1. We explain each step of the algorithm below.

In step 1, we use $\beta = 3C+1$ instead of $2C+1$. This trick is used to avoid certain computing. For example, if $0 < i < \mu_1$, then $ia_1 < \mu_1 a_1 = f(\beta)$. So when we apply Algorithm $\mathbf{UPolySIInt}(ia_1, \beta, x, C)$, it may return $failure$, since with high probability, one of the coefficients is not in $[-C, C]$. On the other hand, this will never happen when $\beta = 2C+1$.

In step 3, we find a lower degree bound d of $f(x)$. k_1, k_2 are the upper bounds of μ_1, μ_2 by Lemma 6. Q_1, Q_2, E are the quantities defined in Lemma 8.

In step 4, by Lemma 8, if $Q_1 \ge E$, then $\frac{\mu_2}{\mu_1} > \frac{|Q_1|}{E} \ge 1$, or $\mu_2 > \mu_1$. If $Q_2 \le \frac{1}{E}$, then $\frac{\mu_2}{\mu_1} < Q_2 E \le 1$, or $\mu_2 < \mu_1$. If both of them are not satisfied, then we just compare the bounds k_1, k_2 of μ_1, μ_2, respectively.

In step 5, we handle the probably case $\mu_2 > \mu_1$. We first use $h(\beta)$ to recover $h(x)$. We need to know the number μ_1. We let i increases from 1 to k_1 and μ_1 is one of them. We check three cases: (1) From Lemma 8, we know $Q_1 \frac{1}{E} < \frac{\mu_2}{\mu_1} < Q_2 E$, so $Q_1 \frac{1}{E} \mu_1 < \mu_2 < Q_2 E \mu_1$. If the interval $(\frac{Q_1}{E} i, Q_2 E i)$ includes an integer, it could be μ_1; if it does not, then i cannot be μ_1. (2) If $f = failure$ or $g = failure$, then we increase i by one. (3) If $h(\beta+1) = \frac{f(\beta+1)}{g(\beta+1)}$, then we return the result. Note that the probabilistic property of the algorithm comes from here: even if $h(\beta+1) = \frac{f(\beta+1)}{g(\beta+1)}$, we are not sure whether we have the correct h. In step 6, we handle the probably case $\mu_1 > \mu_2$, which is similar to step 5.

We now prove the bound of μ. If $Q_1 \ge E$ or $Q_2 \le \frac{1}{E}$, then it is easy to see that $\mu = \min\{\mu_1, \mu_2\}$. So now we assume $Q_1 < E$ and $Q_2 > \frac{1}{E}$. Firstly, we have $E = 1 + \frac{2C}{\beta(\beta-1)} < 1 + \frac{1}{\beta}$ and $Q_1 = Q_2 \frac{\beta^{D-d}}{(\beta+1)^{D-d}}$. Since $Q_1 \frac{1}{E} < \frac{\mu_2}{\mu_1} < Q_2 E$, $Q_2 > \frac{1}{E}$ and $Q_1 < E$, we have $\frac{\mu_2}{\mu_1} > Q_1 \frac{1}{E} = Q_2 \frac{\beta^{D-d}}{(\beta+1)^{D-d}} \frac{1}{E} > \frac{\beta^{D-d}}{(\beta+1)^{D-d}} (\frac{1}{E})^2 > \frac{\beta^{D-d+2}}{(\beta+1)^{D-d+2}}$. So $\mu_1 < \frac{(\beta+1)^{D-d+2}}{\beta^{D-d+2}} \mu_2 < (1+\frac{1}{3C})^{D-d+2} \mu_2$. For the similar reason, we have $\mu_2 < (1+\frac{1}{3C})^{D-d+2} \mu_1$. So we have $\mu \le (1+\frac{1}{3C})^{D-d+2} \min\{\mu_1, \mu_2\}$.

The analysis of arithmetic complexity is similar to that of Theorem 11. Since the missing factor μ may destroy the sparse structure, we use D instead of T. Since $\beta = 3C + 1$, $f(\beta)$ is $\mathcal{O}(C^D)$ and the height of the data is $\mathcal{O}(D \log C)$. □

4 Multivariate Rational Function Interpolation

4.1 Multivariate Polynomial Interpolation with Kronecker Substitution

In this section, we will give an algorithm based on a variant Kronecker substitution, which will be used in the multivariate rational function interpolation.

In the rest of section, we assume that the variables are ordered as $x_1 \prec x_2 \prec \cdots \prec x_n$, and the lexicographic monomial order will be used. Let $m = x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}$ be a monomial and $\beta_1, \beta_2, \cdots, \beta_n \in \mathbb{N}$. Then we denote $\widehat{m} := \beta_1^{k_1} \beta_2^{k_2} \cdots \beta_n^{k_n}$.

Lemma 9. *Let m_1, m_2 be monomials, $\deg_{x_j}(m_i) \le D, j = 1, 2, \cdots, n, i = 1, 2$. If $m_1 > m_2$ in the lexicographic order and $\beta_1 > 1$, $\beta_i \ge \beta_{i-1}^{D+1}, i = 2, \ldots, n$, then $\widehat{m}_1 > \widehat{m}_2$ and $\frac{\widehat{m}_2}{\widehat{m}_1} \le \frac{1}{\beta_1}$.*

Proof. Assume $m_1 = x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}, m_2 = x_1^{s_1} x_2^{s_2} \cdots x_n^{s_n}$. As $m_1 > m_2$, without loss of generality, assume $k_n > s_n$. First we have $\beta_1^{s_1} \beta_2^{s_2} \cdots \beta_n^{s_n} \le \beta_1^D \beta_2^D \cdots \beta_{n-1}^D \beta_n^{k_n - 1}$ and $\beta_n^{k_n} \le \beta_1^{k_1} \beta_2^{k_2} \cdots \beta_n^{k_n}$. It is sufficient to prove $\beta_1^D \beta_2^D \cdots \beta_{n-1}^D \beta_n^{k_n - 1} < \beta_n^{k_n}$. Dividing $\beta_n^{k_n - 1}$ on both sides, it is sufficient to prove $\beta_1^D \beta_2^D \cdots \beta_{n-1}^D < \beta_n$. Since $\beta_1 \cdot \beta_1^D \beta_2^D \cdots \beta_{n-1}^D = \beta_1^{D+1} \beta_2^D \cdots \beta_{n-1}^D \le \beta_2^{D+1} \beta_3^D \cdots \beta_{n-1}^D \le \cdots \le \beta_{n-1}^{D+1} \le \beta_n$, we have $\beta_1 \cdot \beta_1^D \beta_2^D \cdots \beta_{n-1}^D \le \beta_n$. Since $\beta_1 > 1$, $\beta_1^D \beta_2^D \cdots \beta_{n-1}^D < \beta_n$. So we have $\widehat{m}_1 > \widehat{m}_2$. We have proved the first part. Since $\frac{\widehat{m}_2}{\widehat{m}_1} = \frac{\beta_1^{s_1} \beta_2^{s_2} \cdots \beta_n^{s_n}}{\beta_1^{k_1} \beta_2^{k_2} \cdots \beta_n^{k_n}}$, we assume there exists an i such that $k_{i+1} = s_{i+1}, k_{i+2} = s_{i+2}, \ldots, k_n = s_n$, and $k_i > s_i$. Then $\frac{\widehat{m}_2}{\widehat{m}_1} = \frac{\beta_1^{s_1} \beta_2^{s_2} \cdots \beta_i^{s_i}}{\beta_1^{k_1} \beta_2^{k_2} \cdots \beta_i^{k_i}} \le \frac{\beta_1^D \beta_2^D \cdots \beta_{i-1}^D \beta_i^{k_i - 1}}{\beta_1^{k_1} \beta_2^{k_2} \cdots \beta_{i-1}^{k_{i-1}} \beta_i^{k_i}} = \frac{\beta_1^D \beta_2^D \cdots \beta_{i-1}^D}{\beta_i} \cdot \frac{1}{\beta_1^{k_1} \beta_2^{k_2} \cdots \beta_{i-1}^{k_{i-1}}} \le \frac{1}{\beta_1}$. □

Lemma 10. *Let $f = \sum_{i=1}^{t} c_i m_i \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$, $\|f\|_\infty \le C$, $\deg(f) \le D$, $\beta_1 \ge 2C + 1$, and $\beta_i \ge \beta_{i-1}^{D+1}$ for $i = 2, 3, \ldots, n$. Then $f(x)$ can be uniquely determined by $f(\beta_1, \beta_2, \ldots, \beta_n)$.*

Proof. Assume m_t is the leading term and $m_t = x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$. First we show that m_t is unique determined by $f(\beta_1, \beta_2, \ldots, \beta_n)$. Let $A_n = |f(\beta_1, \beta_2, \ldots, \beta_n)|, A_j = \frac{|f(\beta_1, \beta_2, \ldots, \beta_n)|}{\beta_{j+1}^{d_{j+1}} \beta_{j+2}^{d_{j+2}} \cdots \beta_n^{d_n}}, j = 1, 2, \ldots, n-1$. Now we prove that for any $j = 1, 2, \ldots, n$, if $k \le d_j$, then $\frac{A_j}{\beta_j^k} > \frac{1}{2}$. If $k > d_j$, then $\frac{A_j}{\beta_j^k} < \frac{1}{2}$. Since $\beta_j > 1$, it is sufficient to show that if $k = d_j$, then $|\frac{A_j}{\beta_j^k}| > \frac{1}{2}$; if $k = d_j + 1$, then $|\frac{A_j}{\beta_j^k}| < \frac{1}{2}$. First note $\frac{\widehat{m}_i}{\widehat{m}_t} = \prod_{j=i}^{t-1} \frac{\widehat{m}_j}{\widehat{m}_{j+1}} \le \frac{1}{\beta_1^{t-i}}$. When $k = d_j$, we have

$|f(\beta_1,\beta_2,\cdots,\beta_n)| \geq |c_t|\widehat{m}_t - C\sum_{i=1}^{t-1}\widehat{m}_i = \widehat{m}_t(|c_t| - C\sum_{i=1}^{t-1}\frac{\widehat{m}_i}{\widehat{m}_t}) \geq \widehat{m}_t(|c_t| - C\sum_{i=1}^{t-1}\frac{1}{\beta_1^i}) \geq \widehat{m}_t(1-\frac{C}{\beta_1-1}+\frac{C}{\beta_1^t-\beta_1^{t-1}}) > \frac{1}{2}\widehat{m}_t$. So $\frac{A_j}{\beta_j^k} > \frac{1}{2}\frac{\widehat{m}_t}{\beta_j^k\beta_{j+1}^{d_{j+1}}\beta_{j+2}^{d_{j+2}}\cdots\beta_n^{d_n}} \geq \frac{1}{2}$. When $k = d_j+1$, $|f(\beta_1,\beta_2,\cdots,\beta_n)| \leq C\sum_{i=1}^{t}\widehat{m}_i = C\widehat{m}_t(1+\sum_{i=1}^{t-1}\frac{\widehat{m}_i}{\widehat{m}_t}) \leq C\widehat{m}_t(1+\sum_{i=1}^{t-1}\frac{1}{\beta_1^i}) = C\widehat{m}_t\frac{\beta_1-\frac{1}{\beta_1^{t-1}}}{\beta_1-1} = \widehat{m}_t\frac{C}{\beta_1-1}(\beta_1-\frac{1}{\beta_1^{t-1}}) \leq \frac{1}{2}\widehat{m}_t\beta_1 - \frac{1}{2}\widehat{m}_t\frac{1}{\beta_1^{t-1}}$. Clearly, $\frac{\widehat{m}_t\beta_1}{\beta_j^k\beta_{j+1}^{d_{j+1}}\beta_{j+2}^{d_{j+2}}\cdots\beta_n^{d_n}} \leq 1$, so $\frac{A_j}{\beta_j^k} < \frac{1}{2}$. So $d_1,d_2,\ldots,d_n$ can be determined by $f(\beta_1,\beta_2,\cdots,\beta_n)$. Now we show that c_t also can be determined. Let $g = \sum_{i=1}^{t-1}c_im_i$. So $f = g + c_tm_t$. By the above, we have $\frac{|g(\beta_1,\beta_2,\cdots,\beta_n)|}{\widehat{m}_t} < \frac{1}{2}$. So $\frac{f(\beta_1,\beta_2,\cdots,\beta_n)}{\widehat{m}_t} = \frac{g(\beta_1,\beta_2,\cdots,\beta_n)}{\widehat{m}_t} + c_t$, which is $|\frac{|f(\beta_1,\beta_2,\cdots,\beta_n)}{\widehat{m}_t} - c_t| < \frac{1}{2}$. So $c_t = \lfloor\frac{f(\beta_1,\beta_2,\cdots,\beta_n)}{\widehat{m}_t} + \frac{1}{2}\rfloor$. The rest can be proved by induction. □

Lemma 11. *Let* $f = \sum_{i=1}^{t}f_ix_n^{d_i} \in \mathbb{Z}[x_1,x_2,\ldots,x_n]$, $f_i \in \mathbb{Z}[x_1,x_2,\ldots,x_{n-1}]$, $\deg(f) \leq D$, $\|f\|_\infty \leq C$, *and* $\#f \leq T$. *If* $\beta_1 \geq 2C+1, \beta_i \geq \beta_{i-1}^{D+1}, i = 2,3,\ldots,n$, *then* $|f_i(\beta_1,\beta_2,\ldots,\beta_{n-1})| < C\beta_{n-1}^D\frac{\beta_1}{\beta_1-1}, i = 1,2,\ldots,t$.

Proof. It is easy to see that T, D, C are also the corresponding bounds of f_i. Assume that $f_i = \sum_{i=1}^{s}c_im_i$, and $m_1 < m_2 < \cdots < m_s$ in lexicographic order. By Lemma 9, we have $|f_i(\beta_1,\beta_2,\ldots,\beta_{n-1})| \leq C\sum_{i=1}^{s}\widehat{m}_i = C\widehat{m}_s(\sum_{i=1}^{s}\frac{\widehat{m}_i}{\widehat{m}_s}) \leq C\beta_{n-1}^D(1+\sum_{i=1}^{T-1}\frac{1}{\beta_1^i}) = C\beta_{n-1}^D\frac{\beta_1-\frac{1}{\beta_1^{T-1}}}{\beta_1-1} < C\beta_{n-1}^D\frac{\beta_1}{\beta_1-1}$. □

Since $2|f_i(\beta_1,\beta_2,\ldots,\beta_{n-1})| \leq 2C\beta_{n-1}^D\frac{\beta_1-\frac{1}{\beta_1^{T-1}}}{\beta_1-1} < \beta_{n-1}^D\beta_1 \leq \beta_{n-1}^{D+1} \leq \beta_n$, $2|f_i(\beta_1,\beta_2,\ldots,\beta_{n-1})|+1 \leq \beta_n$, we can give the following recursive interpolation algorithm. Note that we regard the upper bound $C \geq \|f\|_\infty$ as a fixed number in the recursive process.

The parameter ρ in the input needs some explanation. If the interpolation is for a polynomial f, then $\rho = f(\beta_1,\beta_2,\ldots,\beta_n)$ and the algorithm will return f. In Algorithm 20, $\rho = \frac{kf(\beta_1,\beta_2,\ldots,\beta_n)}{\mu}$ for some integers k and μ. For $k = \mu$, the algorithm returns f and for $k \neq \mu$ the algorithm may fail.

Algorithm 17 (MPolySIInt)

Input: A list $\beta_1,\beta_2,\ldots,\beta_n$ in $\mathbb{N}$, which satisfy the condition of Lemma 11; $\rho \in \mathbb{Z}$; $T, D \in \mathbb{N}$, where $T \geq \#f, D \geq \deg(f)$.
Output: The exact form of $f(x_1,x_2,\ldots,x_n)$ or failure.

Step 1: **If** $n = 1$, **then** $C_1 := C$, **else** $C_1 := \lfloor C\beta_{n-1}^D\frac{\beta_1}{\beta_1-1}\rfloor$.
Step 2: Let $g :=$ **UPolySIInt**(β_n,ρ,C_1,x_n); **if** ($g = failure$ or $\deg(g) > D$) **then** return $failure$; **end if**; Assume $g = c_1x_n^{d_1} + c_2x_n^{d_2} + \cdots + c_tx_n^{d_t}$.
Step 3: **If** $n = 1$, **then** return g;
Step 4: Let $f := 0$;
for $i = 1,2,\ldots,t$ do
Let $M :=$ **MPolySIInt**$(\beta_1,\beta_2,\ldots,\beta_{n-1},c_i,T-t+1,D-d_i)$. **if** $M =$ failure **then return** failure; **end if**. Let $f := f + Mx_n^{d_i}$;
Step 5: **return** f.

Theorem 18. *The algorithm is correct. The arithmetic complexity is* $\mathcal{O}(nT\log^2 D)$*, and the height of the data is* $\mathcal{O}(D^n \log C)$.

Proof. By Theorem 5, the arithmetic operations of Algorithm **UPolySIInt** are $\mathcal{O}(T\log^2 D)$, and we call n times Algorithm **UPolySIInt**, so the arithmetic operations are $\mathcal{O}(nT\log^2 D)$. The reason for the height of the data is the same as Theorem 5. □

4.2 Probabilistic Multivariate Rational Function Interpolation

In this and the next subsections, we denote $\mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{k} = (k_1, k_2, \ldots, k_n)$, $f(\mathbf{x^k}) = f(x_1^{k_1}, x_2^{k_2}, \cdots, x_n^{k_n})$, $\mathbf{x}+x = (x_1+x, x_2+x, \ldots, x_n+x)$, $f_{\mathbf{c}}(x) = f(x+c_1, (x+c_2)^{2D+1}, \ldots, (x+c_n)^{(2D+1)^{n-1}})$, where $\mathbf{c} = (c_1, c_2, \ldots, c_n)$.

Assume $h = f/g \in \mathbb{Z}(\mathbf{x}), \gcd(f,g) = 1, T \geq \#h, D \geq \deg(h), C \geq \|h\|_\infty$. We first prove a lemma.

Lemma 12. *Assume* $f, g \in \mathbb{Z}[\mathbf{x}]$, $\gcd(f,g) = 1$. *If* $k_1, k_2, \ldots, k_n$ *are any positive numbers, then* $\gcd(f(\mathbf{x^k}), g(\mathbf{x^k})) = 1$.

Proof. Let $h_i = \mathbf{res}(f, g, x_i) = s_i f + t_i g, i = 1, 2, \ldots, n$. From $\gcd(f,g) = 1$, we have $h_i \neq 0$. Replacing x_j by $x_j^{k_j}, j = 1, 2, \ldots, n$, we have $h_i(\mathbf{x^k}) = s_i(\mathbf{x^k}) f(\mathbf{x^k}) + t_i(\mathbf{x^k})$ $g(\mathbf{x^k})$. So $\gcd(f(\mathbf{x^k}), g(\mathbf{x^k}))|h_i(\mathbf{x^k}), i = 1, 2, \ldots, n$. Since $h_i \neq 0$, it is easy to see that $h_i(\mathbf{x^k}) \neq 0$, and we know $h_i(\mathbf{x^k})$ does not contain x_i. So $\gcd(f(\mathbf{x^k}), g(\mathbf{x^k}))$ does not contain $x_i, i = 1, 2, \ldots, n$. So we have $\gcd(f(\mathbf{x^k}), g(\mathbf{x^k})) = 1$. □

Lemma 13. *Let* $f, g \in \mathbb{Z}[\mathbf{x}]$ *and* $\gcd(f,g) = 1$. *Then* $\gcd(f(\mathbf{x}+x), g(\mathbf{x}+x)) = 1$.

Proof. Let $h_i = \mathbf{res}(f, g, x_i) = s_i f + t_i g, i = 1, 2, \ldots, n$. From $\gcd(f,g) = 1$, we have $h_i \neq 0$. Replacing x_j by $x_j + x, j = 1, 2, \ldots, n$, we have $\gcd(f(\mathbf{x}+x), g(\mathbf{x}+x))|h_i(\mathbf{x}+x)$. Since $h_i(\mathbf{x}+x)$ does not contain x_i, $\gcd(f(\mathbf{x}+x), g(\mathbf{x}+x))$ contains variable x only. Denote $u(x) := \gcd(f(\mathbf{x}+x), g(\mathbf{x}+x))$. Then $f(\mathbf{x}+x) = u(x)a, g(\mathbf{x}+x) = u(x)b$, where $a, b \in \mathbb{Z}[x, \mathbf{x}]$. If $u(x)$ is not a nonzero constant, then let $\beta \in \mathbb{C}$ be a root of $u(x)$, and we have $f(\mathbf{x}+\beta) = u(\beta)a(\beta, \mathbf{x}) = 0$. Since the terms not containing variate x in f are the same as the the ones in $(f(\mathbf{x}+\beta))$, $f(\mathbf{x}+\beta) \neq 0$. This is a contradiction. So $\gcd(f(\mathbf{x}+x), g(\mathbf{x}+x)) = 1$. □

Theorem 19. *Let* $f, g \in \mathbb{Z}[\mathbf{x}]$, $\gcd(f,g) = 1$, $D \geq \max\{\deg(f), \deg(g)\}$, $x, c_1, c_2, \ldots, c_n$ *new variables. Then we have* $\gcd(f_{\mathbf{c}}(x), g_{\mathbf{c}}(x)) = 1$.

Proof. By the two lemmas above, we can easily obtain the theorem. □

By Theorem 19, $R = \mathbf{res}(f_{\mathbf{c}}(x), g_{\mathbf{c}}(x), x)$ is a nonzero polynomial about $c_1, c_2, \ldots, c_n$. Then when we randomly choose $c_1, c_2, \ldots, c_n$, with high probability, that $R(c_1, c_2, \ldots, c_n) \neq 0$. So we reduce the multivariate case into univariate case. But the procedure will destroy the sparse structure. In order to avoid this problem, we randomly choose $c_1, c_2, \ldots, c_n$ satisfying $c_1 \leq c_2 \leq \cdots \leq c_n$, and then randomly choose a $\beta \geq 2C+1$ and let $\beta_i = (\beta + c_i)^{(2D+1)^{i-1}}, i = 1, 2, \ldots, n$. Then these $\beta_i, i = 1, 2, \ldots, n$ satisfy the condition of Lemma 10.

Assume $h(\beta_1, \beta_2, \ldots, \beta_n) = \frac{a}{b}$, $\gcd(a,b) = 1$, $a = \frac{f(\beta_1, \beta_2, \ldots, \beta_n)}{\mu}, b = \frac{g(\beta_1, \beta_2, \cdots, \beta_n)}{\mu}$.

Lemma 14. *Suppose* $h = \frac{f}{g} \in \mathbb{Z}(\mathbf{x})$. *Let* $c_1 \le c_2 \le \cdots \le c_n$ *be positive integers such that* $\gcd(f_{\mathbf{c}}(x), g_{\mathbf{c}}(x)) = 1$, *and* $\beta \ge 2TC^2 + 1$. *Then there exists a unique* $h(x_1, x_2, \ldots, x_n)$ *with* $C \ge \|h\|_\infty$ *corresponding to* $h_{\mathbf{c}}(\beta)$.

Proof. When $\gcd(f_{\mathbf{c}}(x), g_{\mathbf{c}}(x)) = 1$, $h(\mathbf{x})$ is in one-to-one correspondence with $h_{\mathbf{c}}(x)$. Assume there exists another rational function $\frac{f_1(\mathbf{x})}{g_1(\mathbf{x})}$ with $C \ge \|\frac{f_1}{g_1}\|_\infty$ such that $\frac{f_{\mathbf{c}}(\beta)}{g_{\mathbf{c}}(\beta)} = \frac{f_{1\mathbf{c}}(\beta)}{g_{1\mathbf{c}}(\beta)}$, which can be changed into $f_{\mathbf{c}}(\beta)g_{1\mathbf{c}}(\beta) = g_{\mathbf{c}}(\beta)f_{1c}(\beta)$. Define $w(\mathbf{x}) := f(\mathbf{x})g_1(\mathbf{x})$ and $v(\mathbf{x}) := f_1(\mathbf{x})g(\mathbf{x})$. Then $w_{\mathbf{c}}(\beta) = v_{\mathbf{c}}(\beta)$. Since $2D \ge \deg(w), \deg(v)$, $TC^2 \ge \|w\|_\infty, \|v\|_\infty$, by Lemma 10, we have $w(\mathbf{x}) = v(\mathbf{x})$, so $\frac{f(\mathbf{x})}{g(\mathbf{x})} = \frac{f_1(\mathbf{x})}{g_1(\mathbf{x})}$, and the lemma is proved. □

Now we can give a probability algorithm.

Algorithm 20 (MRFunSI1)

Input: A black box $h = \frac{f}{g} \in \mathbb{Z}(\mathbf{x})$, D, T, C, N, where $D \ge \deg(h), T \ge \#h, C \ge \|h\|_\infty$, N is a big positive integer.
Output: The exact form of $h(\mathbf{x})$ or a wrong rational function.

Step 1: Let $\beta = 2TC^2 + 1$. Randomly choose $c_1, c_2, \ldots, c_n \in \{1, 2, \ldots, N\}$ such that $c_1 \le c_2 \le \cdots \le c_n$. Let $\beta_i = (\beta + c_i)^{(2D+1)^{i-1}}, i = 1, 2, \ldots, n$.
Step 2: Evaluate $h(\beta_1, \beta_2, \cdots, \beta_n) = \frac{a}{b}$, where $\gcd(a, b) = 1$.
Step 3: $i = 1$;
Step 4: Let $f =$ **MPolySIInt**$(\beta_1, \beta_2, \ldots, \beta_n, a \cdot i, T, D, C)$;
if $f = failure$ **then** $i := i + 1$; go to Step 4; **end if**
Step 5: Let $g =$ **MPolySIInt**$(\beta_1, \beta_2, \ldots, \beta_n, b \cdot i, T, D, C)$;
if $g = failure$ **then** $i := i + 1$; go to Step 4; **end if**
Step 6: **Return** $\frac{f}{g}$.

Theorem 21. *The algorithm is correct. The arithmetic complexity is* $\mathcal{O}(\mu n T \log^2 D)$, *and the height of the data is* $\mathcal{O}((2D)^n \log(TC^2 + N))$.

Proof. By Lemma 14, if $\gcd(f_{\mathbf{c}}(x), g_{\mathbf{c}}(x)) = 1$, then we can find a rational function with coefficients bounded by C only when $i = \mu$. So in this case, the algorithm returns a correct h. Otherwise, it may return a wrong rational function.

By Theorem 18, the arithmetic complexity of Algorithm **MPolySIInt** is $\mathcal{O}(nT \log^2 D)$. The algorithm calls algorithm **MPolySIInt** at most μ times, so the arithmetic complexity is $\mathcal{O}(\mu n T \log^2 D)$. The reason for the height of the data is the same as Theorem 5. In this case, the degree is $\mathcal{O}((2D)^n)$, and β is $\mathcal{O}(TC^2 + N)$. So the height of the data is $\mathcal{O}(D^n \log(TC^2 + N))$. □

We now analyze the successful rate of Algorithm 20.

Lemma 15. *Let* R *be an integral domain,* $S_1, S_2, \ldots, S_n \subseteq R$ *finite sets with* $N = \#S_i, i = 1, 2, \ldots, n$, *and* $r \in R[\mathbf{x}]$ *a polynomial of total degree at most* $d \in \mathbb{N}$. *If* r *is not the zero polynomial, then* r *has at most* dN^{n-1} *zeros in* $S_1 \times S_2 \times \cdots \times S_n$.

Proof. We prove it by induction on n. The case $n = 1$ is clear, since a nonzero univariate polynomial of degree at most d over an integral has at most d zeros. For the induction step, we write r as a polynomial in x_n: $r = \sum_{0 \le i \le k} r_i x_n^i$ with $r_i \in R[x_1, x_2, \ldots, x_{n-1}]$ for $0 \le i \le k$ and $r_k \neq 0$. Then $\deg(r_k) \le d - k$. By the induction hypothesis, r_k has at most $(d-k)N^{n-2}$ zeroes in $S_1 \times S_2 \times \cdots \times S_{n-1}$. So that there are at most $(d-k)N^{n-1}$ common zeroes of r and r_k in $S_1 \times S_2 \times \cdots \times S_n$. Furthermore, for each $a \in S_1 \times S_2 \times \cdots S_{n-1}$ with $r_k(a) \neq 0$, the univariate polynomial $r_a = \sum_{0 \le i \le k} r_i(a) x_n^i \in R[x_n]$ of degree k has at most k zeros, so that the total number of zeros of r in S^n is bound by $(d-k)N^{n-1} + kN^{n-1} = dN^{n-1}$. □

Theorem 22. *$S_1, S_2, \ldots, S_n$ are n different positive integer sets with $\#S_i = N$. Assume $a_i < a_j$ for $i < j$ and any elements $a_i \in S_i$ and $a_j \in S_j$. If $c_1, c_2, \ldots, c_n$ are randomly chosen in $S_1 \times S_2 \cdots \times S_n$, then Algorithm 20 returns the correct result with probability at least $1 - \frac{2(2D+1)^{2n}}{N}$.*

Proof. By Lemma 14, when $\gcd(f_{\mathbf{c}}(x), g_{\mathbf{c}}(x)) = 1$ in the above algorithm, we obtain the correct result. By Theorem 19, we know $\mathbf{res}(f_{\mathbf{c}}(x), g_{\mathbf{c}}(x), x) \neq 0$. We can see $\deg_x f_{\mathbf{c}}(x) < (2D+1)^n, \deg_x g_{\mathbf{c}}(x) < (2D+1)^n$, $\deg_{\mathbf{c}} f_{\mathbf{c}}(x) < (2D+1)^n, \deg_{\mathbf{c}} g_{\mathbf{c}}(x) < (2D+1)^n$, so $\deg_{\mathbf{c}} \mathbf{res}(f_{\mathbf{c}}(x), g_{\mathbf{c}}(x), x) < 2(2D+1)^{2n}$. By Lemma 15, if $c_1, c_2, \ldots, c_n$ are randomly chosen from $S_1 \times S_2 \cdots \times S_n$, then the probability of resultant polynomial be zero at point $(c_1, c_2, \ldots, c_n)$ is no more than $\frac{2(2D+1)^{2n}}{N}$. So the success rate of Algorithm 20 is at least $1 - \frac{2(2D+1)^{2n}}{N}$. □

5 Experimental Results

The algorithms are implemented in Maple and their practical performances will be presented in this section. The data are collected on a desktop with Windows system, 3.60 GHz Core $i7 - 4790$ CPU, and 8 GB RAM memory. The Maple codes can be found in

http://www.mmrc.iss.ac.cn/~xgao/software/siratfunc.zip

Five randomly constructed rational functions are used to obtain the average times. We have three groups of experiments to present. The first and second groups are about univariate rational function interpolation. The third group is about multivariate rational function interpolation. We use some tricks in our implementations to improve the efficiency, which can be found in the arXiv version of this paper [4].

In Figs. 1 and 2, we compare the two deterministic algorithms **URFunSI1** and **URFunSI2** for univariate rational function interpolation. By the **Base Case**, we mean the sum of the times of interpolating f and g separately. From the data, we can see that: (1) the algorithm using two points are faster than that using one point and (2) the times for interpolating $h = \frac{f}{g}$ are almost the same as that of interpolating f and g, which means that our interpolation algorithm for univariate rational functions are almost optimal.

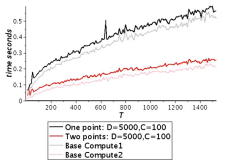


Fig. 1. Univariate: average running times with varying T

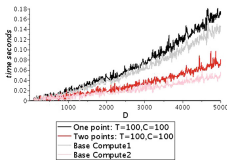


Fig. 2. Univariate: average running times with varying D

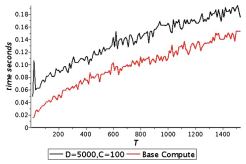


Fig. 3. URFunSIP: average running times with varying T

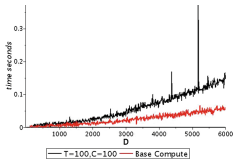


Fig. 4. URFunSIP: average running times with varying D

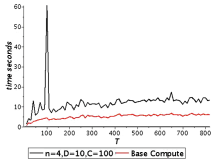


Fig. 5. MRFunSIP1: average running times with varying T

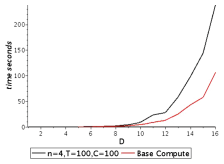


Fig. 6. MRFunSIP1: average running times with varying D

In Figs. 3 and 4, we present the practical performance for the probabilistic algorithm for univariate rational functions. We compare it with the base case. Comparing Figs. 1 and 3, we can see that the probabilistic algorithm is faster than the one point deterministic algorithm and comparable with the two points deterministic algorithm.

For the multivariate algorithm, in Figs. 5 and 6, we present the practical performances. We also give the time which is the sum of the times of interpolating f and g from $h = \frac{f}{g}$ for comparison. We can see that the algorithm is less sensitive to T and quite sensitive to D. But unlike the univariate case, the interpolation of the multivariate rational function is much more difficult than interpolating its denominator and numerator separatively.

6 Conclusion

In this paper, we consider interpolation of sparse rational functions under the assumption that their coefficients are integers with a given bound. This assumption allows us to recover the rational function $h = f/g$ from evaluations of h at one "large" sample point. Experimental results show that the univariate interpolation algorithm is almost optimal, while the multivariate interpolation algorithm needs further improvements. The main problem is that the sample data is of exponential size in n, due to the use of Kronecker type substitution.

References

1. Cuyt, A., Lee, W.S.: Sparse interpolation of multivariate rational functions. Theor. Comput. Sci. **412**(16), 1445–1456 (2011)
2. Grigoriev, D., Karpinski, M., Singer, M.F.: Computational complexity of sparse rational interpolation. SIAM J. Comput. **23**(1), 1–11 (1994)
3. Huang, Q.-L., Gao, X.-S.: Sparse polynomial interpolation with finitely many values for the coefficients. In: Gerdt, V.P., Koepf, W., Seiler, W.M., Vorozhtsov, E.V. (eds.) CASC 2017. LNCS, vol. 10490, pp. 196–209. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66320-3_15
4. Huang, Q.L., Gao, X.S.: Sparse rational function interpolation with finitely many values for the coefficients (2017). arXiv:1706.00914
5. Huang, Q.L., Gao, X.S.: Sparse interpolation of black-box multivariate polynomials using Kronecker type substitutions (2017). arXiv:1710.01301
6. Kaltofen, E.: Greatest common divisors of polynomials given by straight-line programs. J. ACM **35**, 231–264 (1988)
7. Kaltofen, E., Trager, B.: Computing with polynomials given by black boxes for their evaluations: greatest common divisors, factorization, separation of numerators and denominators. J. Symbolic Comput. **9**, 301–320 (1990)
8. Kaltofen, E., Yang, Z.: On exact and approximate interpolation of sparse rational functions. In: Proceedings of ISSAC 2007. ACM Press, pp. 203–210 (2007)
9. Kronecker, L.: Grundzüge einer arithmetischen Theorie der algebraischen Grössen. J. für die Reine und Angewandte Mathematik **92**, 1–122 (1882)
10. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge University Press, New York (1999)

Virtual Theories – A Uniform Interface to Mathematical Knowledge Bases

Tom Wiesing[1(✉)], Michael Kohlhase[1], and Florian Rabe[2]

[1] FAU Erlangen-Nürnberg, Erlangen, Germany
tom.wiesing@fau.de
[2] Jacobs University Bremen, Bremen, Germany

Abstract. To support mathematical research, engineering, and education by computer systems, we need to deal with the differences between mathematical content collections and information systems available today. Unfortunately, these systems – ranging from Wikipedia to theorem prover libraries are usually only accessible via a dedicated web information system or a low-level API at the level of the raw database content. What we would want is a "programmatic, mathematical API" which would give access to the knowledge-bases programmatically via their mathematical constructions and properties.

This paper takes a step into this direction by interpreting large knowledge bases as OMDoc/MMT theories – modular representations of mathematical objects and their properties. For this, we generalize OMDoc/MMT theories to "virtual theories" – theories so big that they do not fit into main memory – and update its knowledge management algorithms so that they can work directly with objects stored in external knowledge bases. An additional technical contribution is the introduction of a codec system that bridges between low-level encodings in databases and the abstract construction of mathematical objects.

1 Introduction

There are various large-scale sources of mathematical knowledge. These include

- generic information systems like Wikipedia,
- collections of informal but rigorous mathematical documents – e.g. research libraries, publisher's "digital libraries", or the Cornell preprint arXiv,
- literature information systems like zbMATH or MathSciNet,
- databases of mathematical objects – like the GAP group libraries, the Online Encyclopedia of Integer sequences (OEIS [Slo03, OEIS]), and the L-Functions and Modular Forms Database (LMFDB [Cre16, LMFDB]),
- fully formal theorem prover libraries like those of Mizar, Coq, PVS, and the HOL systems.

We will use the term **mathematical knowledge bases** to refer to them collectively and restrict ourselves to those that are available digitally. They are very

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 243–257, 2017.
https://doi.org/10.1007/978-3-319-72453-9_17

useful in mathematical research, applications, and education. Commonly these systems are only accessible via a dedicated web interface that allows humans to query or browse the databases. A programmatic interface, if it exists at all, is usually system specific, to use it, users need to be familiar both with the mathematical background and internal structure of the system in question. No predominant standard exists, and these interfaces usually only expose the low-level raw database content. We claim that mathematicians and other scientists desire a "programmatic, mathematical API" that gives access to the knowledge-bases programmatically via their mathematical constructions and properties. We focus on addressing this problem in this paper.

For our implementation we interpret mathematical knowledge bases as OMDoc/MMT theory graphs – modular, flexi-formal representations of mathematical objects, their properties, and relations. This embedding gives us a common conceptual framework to handle different knowledge sources, and the modular and heterogeneous nature of OMDoc/MMT theory graph can be used to reconcile differing ontological commitments of the knowledge sources with in this conceptual framework.

To cope with the scale of common mathematical knowledge bases we generalize OMDoc/MMT theories to "virtual theories", which allow for unlimited, dynamically growing number of declarations. We also update the knowledge management algorithms in the MMT system so that they can directly deal with the databases underlying the knowledge bases. Here we provide a systematic solution for encoding/decoding between low-level representations in standard databases and high-level mathematical representations.

This paper proceeds as follows: In Sect. 2 we give a short overview of OMDoc/MMT theory graphs along with the Math-In-The-Middle approach developed in the OpenDreamKit project, our primary use-case for virtual theories. Section 3 discusses LMFDB and its interface as an example of a very large state-of-the-art mathematical knowledge base, and Sect. 4 shows how it can be represented as a set of virtual theories. Section 5 introduces the codec architecture and describes how to access virtual theories at the semantic/mathematical level, and Sect. 6 makes QMT queries aware of virtual theories. Section 7 concludes the paper.

2 Virtual Research Environments for Mathematics: The Math-in-the-Middle Approach

The work reported in this paper originates from in the EU-funded OpenDreamKit [ODK] project that aims to create virtual research environments (VRE) enabling mathematicians to make efficient use of existing open-source mathematical knowledge systems. These systems include computer algebra systems like SageMath and GAP as well as mathematical data bases such as the LMFDB, which must be made interoperable for integration into a VRE. In the OpenDreamKit project we have developed the Math-in-the-Middle (MitM) approach, which posits a central ontology of mathematical knowledge, which acts as a pivot point

for interoperability; see [Deh+16] for a description of the approach and [Koh+17] for a technical refinement and large-scale interoperability case study.

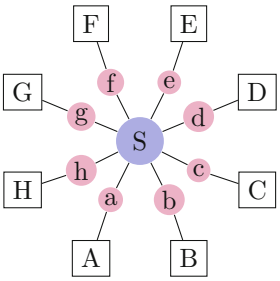


Fig. 1. The MiTM approach to connecting systems.

The MitM ontology in the center of Fig. 1 models the true, underlying mathematical semantics in OMDoc/MMT and allows translation between this centrally formalized knowledge and the systems on the boundary via views and alignments. This mathematical knowledge is modeled using the well-established theory graph paradigm and is stored inside our OMDoc/MMT-based MathHub system [MH].

The knowledge in the mathematical software systems – denoted by square boxes in Fig. 1 – also modeled via OMDoc/MMT theory graphs the **API theories** – the corresponding red circles; these are generated from the knowledge bases in the systems by a custom process. The API theories allow us to implement translation with the help of OMDoc/MMT views and alignments between the ontology – the Math-In-The-Middle – and each of the systems and use these translations for transporting computational tasks between the systems.

The realization of the MitM approach crucially depends on the information architecture of the OMDoc/MMT language [Koh06,RK13] and its implementation in the MMT system [Rab13,MMT].

In OMDoc/MMT knowledge is organized in **theories**, which contain information about mathematical concepts and objects in the form of **declarations**. Theories are organized into an "object-oriented" inheritance structure via **inclusions** and **structures** (for controlled multiple inheritance), which is augmented via truth-preserving mappings between theories called **views**, which allow to relate concepts of pre-existing theories and transport theorems between these. Inclusions, structures, and views impose a graph structure on the represented mathematical knowledge, called a **theory graph**.

We observe that even very large mathematical knowledge spaces about abstract mathematical domains can be represented by small, but densely connected, theory graphs, if we make all inherited material explicit in a process called **flattening**. The OMDoc/MMT language provides systematic names (MMT URIs) for all objects, properties, and relations in the induced knowledge space, and given the represented theory graph, the MMT system can compute them on demand.

Generally, knowledge in a knowledge space given by a theory graph loaded by the MMT system can be accessed by either giving it's MMT URI, or by uniquely describing it via a set of conditions. To achieve the latter, MMT has a Query Language called QMT [Rab12], which allows even complex conditions to be specified. Currently, the MMT system loads the theory graph into main memory at startup and interleaves incremental flattening and query evaluation operations on the MMT data structures until the result has been produced.

In [Koh+17] we show that the MitM approach, its OMDoc/MMT-based realization, and distribution via the SCSCP protocol are sufficient for distributed, federated computation between multiple computer algebra systems (Sage, GAP, and Singular), and that the MitM ontology of abstract group theory can be represented in OMDoc/MMT efficiently. This setup is effective because

- the knowledge spaces behind abstract and computational mathematics can be represented in theory graphs very space-efficiently: The compression factors between a knowledge space and its theory graph – we call it the **TG factor** – exceeds two orders of magnitude even for small domains.
- only small parts of the knowledge space are traversed for a given computation.

But the OpenDreamKit VRE must also include mathematical data sources like the LMFDB or the OEIS, which contain millions of mathematical objects. For such knowledge sources, the classical MMT system is not yet suitable:

- the knowledge space corresponding to the data base content cannot be compressed by "general mathematical principles" like inheritance. Indeed, redundant information is already largely eliminated by the data base schema and the "business logic" of the information system it feeds.
- typically large parts of the knowledge space need to be traversed to obtain the intended results to queries.

Therefore, we extend the concept of OMDoc/MMT theories – which carry the implicit assumption of containing only a small number of declarations (see [FGT92] for a discussion) – to **virtual theories**, which can have an unlimited (possibly infinite) number of declarations. To contrast the intended uses we will call the classical OMDoc/MMT theories **concrete theories**. In practice, a virtual theory is represented by concrete approximations: OMDoc/MMT works with a concrete theory, whose size changes dynamically as a suitable backend infrastructure generates declarations on demand.

3 Example: The API and Structure of LMFDB

The "L-Functions and Modular Forms Database" (LMFDB [LMFa]) is a large database, storing among other mathematical objects several thousand L-Functions and curves along with their properties. Technically, it uses a MongoDB database with a Python web frontend. We use this as an example of a virtual theory. Before we go into this in more detail, we have a closer look at the structure and existing APIs to of LMFDB.

3.1 The Structure of LMFDB

LMFDB has several sub-databases, e.g., for elliptic curves or transitive groups. Within each of these, every object is stored as a single JSON record. Figure 2 shows an example: each property of this JSON object corresponds to a property

```
{
    "degree": 1,
    "x−coordinates_of_integral_points": "[5,16]",
    "isogeny_matrix": [[1,5,25],[5,1,5],[25,5,1]],
    "label": "11a1",
    "_id": "ObjectId('4f71d4304d47869291435e6e')",
    ...
}
```


Fig. 2. Part of an elliptic curve in LMFDB (some fields omitted for brevity)

of the underlying mathematical object. For example, the `degree` property – here 1 – of the JSON objects corresponds to the degree of the underlying elliptic curve.

Other properties are more complex: the value of the `isogeny_matrix` property is a list of lists representing a matrix. This disconnect between JSON encoding and mathematical meaning can become much more severe, e.g., the `x-coordinates_of_integral_points` field is semantically a list of integers but (due to the sizes limits on integers) is encoded as a string.

3.2 An API for LMFDB Objects

Querying is an important application for mathematical knowledge bases. The LMFDB API [Lmf] exposes a querying interface that can be used either by humans via the web or programmatically via JSON-based GET requests over HTTP. A screenshot of the former interface can be seen in Fig. 3.

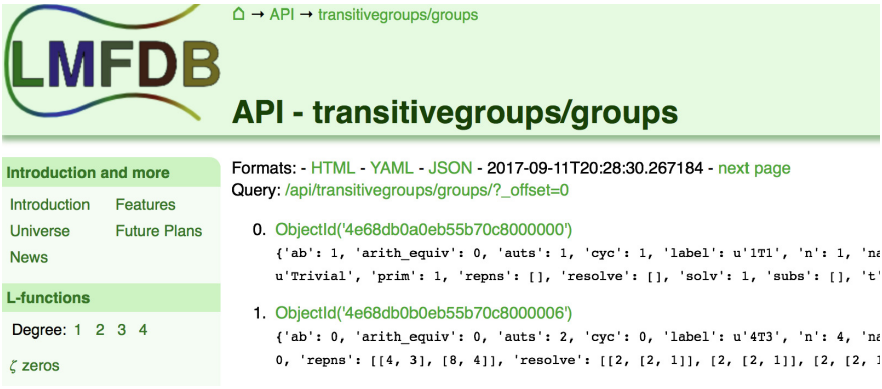


Fig. 3. The web-interface for the LMFDB API.

Queries must name the sub-database to be queried and consist of a set of key-value pairs that correspond to an SQL `where` clause. However, while LMFDB offers a programmable API for accessing its contents, this API sits at the level

of the underlying MongoDB, and not the level of mathematical objects. For example, to retrieve all Abelian objects in the subdatabase of transitive groups, we expect to use the key-value pair `commutative = true`. However, these values need to be encoded to be understood by MongoDB. We need to realize that the database schema actually uses the key `ab` for commutativity, that it has boolean values, and that the schema encodes `true` as `1`. Thus, the actual query to send is http://www.lmfdb.org/api/transitivegroups/groups/?ab=1.

In this example, all steps are relatively straightforward. But in general, e.g. when searching for all elliptic curves with a specific isogeny matrix, this not only requires good familiarity with the mathematical background but also with the system internals of the particular LMFDB sub-database; a skill set commonly found in neither research programmers nor average mathematicians.

Our diagnosis is that LMFDB – and most other mathematical knowledge databases – suffer from two problems:

- *human/computer mismatch:* humans have problems interacting with LMFDB programmatically, because they must speak the system language instead of mathematical language.
- *computer/computer mismatch:* mathematical computer systems cannot interoperate with LMFDB without extending their code, because their system languages differ.

Using the MitM approach we have presented in Sect. 2, we can solve both problems at the same time by lifting the communication to the level of OMDoc/MMT-encoded MitM objects, which both MitM-compatible software systems and humans can understand.

4 LMFDB as a Set of Virtual Theories

The mathematical software systems to be integrated via the MitM approach have so far been computation-oriented, e.g., computer algebra systems. Their API theories typically declare types and functions on these types (the latter including constants seen as nullary functions). Even though database systems differ drastically from these in many respects, they are very similar at the MitM level: a database like LMFDB defines

- some types: each table's schema is essentially one type definition,
- many constants: each table entry is one constant of the corresponding type.

Thus, we can apply essentially the same approach. In particular, the API theories must contain definitions of the database schemas.

From a system perspective, virtual theories behave just like concrete theories, but without the assumption of being able to load all declarations from some file on disk at once. Instead, virtual theories load declarations in a lazy fashion when they are needed. MMT stores concrete theories as XML files. Because most external knowledge bases use databases with low-level APIs, we must allow virtual theories to be stored in external database. Apart from standard software engineering tasks, this leaves three conceptual problems we had to solve:

P1. Turn the database schemas and tables into OMDoc/MMTtheories and declarations.

P2. Lift data in *physical* representation (as records of the underlying database) to OMDoc/MMTobject in *semantic* representation.

P3. Translate semantic queries to queries about physical representations so that they can be executed directly on the database without loading the entire theory into MMT.

We deal with **P1** here, with **P2** in Sect. 5, and with **P3** in Sect. 6.

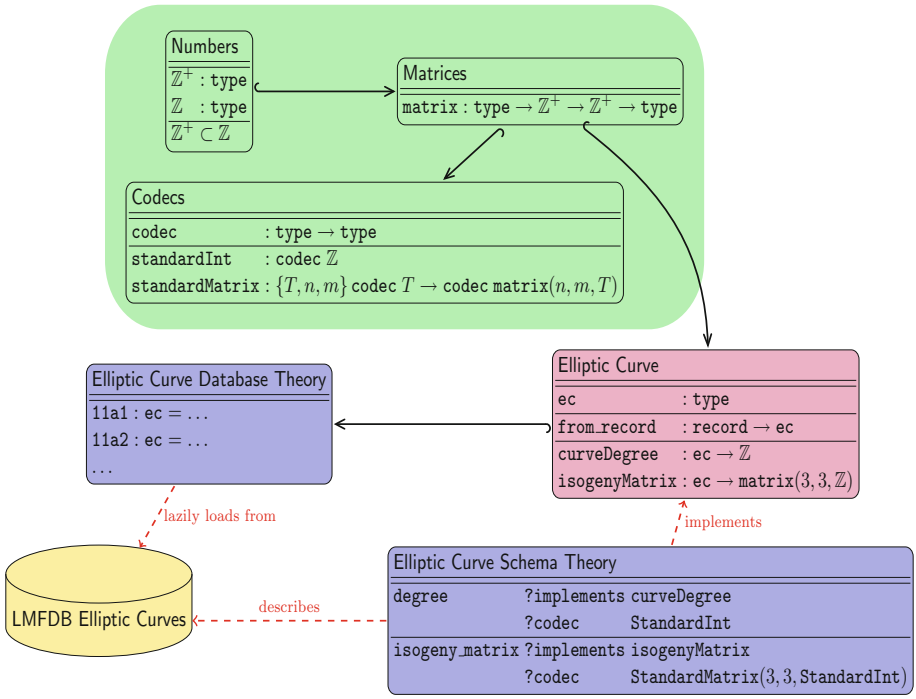


Fig. 4. Virtual theory for LMFDB elliptic curves (some declarations omitted) (Color figure online)

A sketch of our overall solution is given in Fig. 4. The math in the middle comprises preexisting formalizations of general mathematics, here numbers and matrices (in green), and novel LMFDB-specific ones, here elliptic curves (in red). Moreover, we introduce a specification of various codecs to translate between physical and semantic representations. The remaining theories (in blue) form the LMFDB API theories: the schema theory and the database theory, which we describe below.

The set of constants in a database table – while finite – can be arbitrarily large. In particular, all LMFDB tables[1] are just finite subsets of infinite sets, whose size is not limited by mathematical specifications but by computational power: the database holds all objects that users have computed so far and grows constantly as more objects are computed. LMFDB tables usually include a naming system that defines unique identifiers (which are used as the database keys) for these objects, and these identifiers are predetermined even for those objects that have not been computed yet. Thus, it is not practical to fix a set of concrete API theories. Instead, the API theories must be split into two parts: for each database table, we need

- a concrete theory called the **schema theory** that defines the schema and other relevant information about the type of objects in the table and
- a virtual theory called the **database theory** that contains one definition for each value of that type (using the LMFDB identifier as the name of the defined constant).

LMFDB's technical realization does not require formalizing the schema of each table. Instead, the tables are generated systematically and therefore follow an implicit schema that can – in principle – be obtained from the documentation or reverse-engineered from the tables. However (and here LMFDB critically differs from, e.g. the OEIS), the mathematics involved in the tables is so deep that this is not possible in practice for all but a few experts. Therefore, we sat down with the original author of one of the best-documented tables – John Cremona for the table of elliptic curves – and formalized the corresponding schema in OMDoc/MMT.

In the following, we will use this table as a running example. Our methods extend immediately to any other table once its schema has been formalized.

Our formalization models elliptic curves in a very simple fashion by using an abstract type `ec`. The constructor `from_record` takes an MMT record and returns an elliptic curve. Properties of elliptic curves are formalized as functions out of this type. We list only two here as examples: the degree, an integer, and the isogeny matrix, a 3×3 matrix of integers. We omit the relevant axioms, which are not essential for our purposes here. Recall that the Math-in-the-Middle approach models mathematical knowledge "in the middle" independent of any particular system. This is exactly the case here – the model of elliptic curves does not rely on LMFDB, nor any other system, so that we can integrate other knowledge sources about elliptic curves or to future versions of the LMFDB with changed structure.

[1] Technically, LMFDB is implemented using MongoDB and comprises a set of sets (each one called a database) of JSON objects. However, due to the conventions used, we can also understand it conceptually as a set of tables of a relational database, keeping in mind that every row is a tuple of arbitrary JSON objects.

5 Accessing Virtual Theories

We now address **P2**: lifting physical to semantic representations. Intuitively, it is straightforward how to implement a virtual theory V: we use an initially empty concrete theory C, and whenever an identifier id of V is requested, MMT dynamically adds the corresponding declaration of id to C. MMT already abstracts from the physical realizations of persistent storage using the *backend* interface: essentially a backend is any component that allows loading declarations. Thus, we only have to implement a new backend that connects to LMFDB, retrieves the JSON object with identifier id, and turns it into an OMDoc/MMT declaration.

However, this glosses over a major problem: the databases used for the scalable physical storage of large datasets usually offer only very simple data structures. For example, a JSON database (as underlies LMFDB) offers only limited-precision integers, boolean, strings, lists, and records as primitive objects and does not provide a type system. Consequently, the objects stored in the database are very different from the sophisticated mathematical objects expected by the schema theory. Therefore, databases like LMFDB must encode this complex mathematical objects as simple database objects.

5.1 Concrete Encodings of Mathematical Objects

Consider, for example, the field `degree` from Fig. 2 above. Its *semantic* type in the MitM-formalization is $\mathbb{Z}$. However, its *physical* type in LMFDB is `IEEE754` a mixture of 64-bit floating point numbers and strings: integers that exceeds $2^{53} - 1$ are stored as JSON strings containing the corresponding decimal representation. We speak of *encoding* when translating semantic objects to their physical representations and of *decoding* in the dual case, and we speak of *codecs* when referring to a pair of an encoding and a decoding function.

To formally specify codecs, we introduce a new OMDoc/MMT theory `Codecs` as a part of the MitM ontology. Our codecs are indexed by semantic types: the type constructor `codec` maps a semantic type to a new type of codecs for it. For instance, the object `StandardInt` of type `codec` $\mathbb{Z}$ is a codec that translates between LMFDB's idiosyncratic float/string-representation and MitM's integers. Note that there can be multiple different codecs for the same semantic type. For example, `IntAsArray` encodes integers x as lists of 64-bit integers consisting of the digits of x with respect to base 2^{64}.

We do not (and do not have to) define the actual encoding/decoding functions in OMDoc/MMT. It is more important to identify the codecs needed in practice, introduce names for them, and spell out their semantics. Then it is straightforward to implement them in any other programming language used interfacing with LMFDB.

In particular, we have implemented them in Scala, the language underlying the MMT system. Additionally, the Codecs theory annotates each codec declaration with a reference to the Scala class implementing the codec. That way, MMT can run the encoding/decoding functions of the codec.

Codecs		
`codec`	: `type` → `type`	
`StandardPos`	: `codec` $\mathbb{Z}^+$	JSON number if small enough, else JSON string of decimal expansion
`StandardNat`	: `codec` $\mathbb{N}$	
`StandardInt`	: `codec` $\mathbb{Z}$	
`IntAsArray`	: `codec` $\mathbb{Z}$	JSON List of Numbers
`IntAsString`	: `codec` $\mathbb{Z}$	JSON String of decimal expansion
`StandardBool`	: `codec` $\mathbb{B}$	JSON Booleans
`BoolAsInt`	: `codec` $\mathbb{B}$	JSON Numbers 0 or 1
`StandardString`	: `codec` $\mathbb{S}$	JSON Strings

Fig. 5. Codecs specified in MMT ($\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Z}^+$ are as usual, $\mathbb{B}$ are booleans, and $\mathbb{S}$ are Unicode strings)

The above is only sufficient for atomic semantic types, which typically correspond to one (or more) atomic codecs. Consider now the field `isogeny_matrix` of elliptic curves. The semantic representation of one possible value (namely for the curve `11a1`) of this field is the matrix on the right.

$$M = \begin{pmatrix} 1 & 5 & 25 \\ 5 & 1 & 5 \\ 25 & 5 & 1 \end{pmatrix}$$

The semantic type operator Matrix takes 1 type argument (the element type, integers in this case) and two value arguments (the dimensions, 3 and 3 in this case) and constructs the respective matrix type. In principle, one could give a codec for each matrix type that comes up in a database schema. But a much more elegant solution is to specify **codec operator**s in analogy to type operators. A codec operator for a type operator with k type and l value arguments, takes k codec and l value arguments. For example, a codec operator for matrices takes a codec C : `codec` E for the element type E and the dimensions m and n and returns a codec of type `codec` (Matrix $E\, m\, n$).

Codecs (continued)		
`StandardList`	: $\{T\}$ `codec` $T \to$ `codec` List(T)	JSON list, recursively coding each element of the list
`StandardVector`	: $\{T, n\}$ `codec` $T \to$ `codec` Vector(n, T)	JSON list of fixed length n
`StandardMatrix`	: $\{T, n, m\}$ `codec` $T \to$ `codec` Matrix(n, m, T)	JSON list of n lists of length m

Fig. 6. Second annotated subset of the codecs theory containing a selection of codec operators found in MMT. Compare with Fig. 5.

Like codecs, codec operators are represented in MMT in two ways: as declarations inside the theory `Codecs` (see Fig. 6 for a list, compare again with Fig. 4) and as a corresponding Scala function that maps codecs to codecs. When

reading the declarations, note that we make use of the dependent function types of the MitM foundation: curly brackets denote dependent function arguments, i.e., arguments that may occur in later argument types and the result type.

With these declarations, we recover the LMFDB encoding of isogeny matrices by applying the codec operator `StandardMatrix`, which encodes matrices as lists of lists, to the codec `StandardInt` and the dimension 3 and 3. The resulting codec `StandardMatrix`$(\mathbb{Z}, 3, 3,$ `StandardInt`$)$ encodes the above matrix as [[1.0,5.0,25.0],[5.0,1.0,5.0],[25.0,5.0,1.0]].

5.2 Choosing Encodings in Schema Theories

If we ignore encoding issues, schema theories are straightforward: they contain one declaration of the same name for each field within an LMFDB record. This specifies only the semantic type of each field and does not relate it to the MitM formalization. To handle the encoding as a physical type, we annotate each declaration with the codec that the databases for the values of that field. Moreover, to connect the schema theories to the MitM formalization, we additionally annotate each field with the corresponding property of elliptic curves from the MitM theory. We can now understand the last unexplained parts of Fig. 4. `?implements` is the symbol used to annotate the metadatum, which MitM property a schema field corresponds to. And `?codec` similarly annotates the codec to each field.

For example, the `degree` field implements the `curveDegree` property in the elliptic curve theory and uses the `StandardInt` codec. Thus, the schema theories determine the entire relation between semantic and physical objects.

The database theory is a virtual theory and contains one declaration per LMFDB record. Given the URI of an object in the respective database, our MMT backend for LMFDB first retrieves the appropriate record from LMFDB – in the case of `11a1` this corresponds to retrieving the JSON found in Fig. 2. Then, for each field, it uses the annotated codec (which is an OMDoc/MMT expression) to build an actual codec (as a runnable Scala function) and runs its decoding function. Next, it passes the resulting record to the `from_record` constructor, which yields an elliptic curve in the MitM theories. Finally, this elliptic curve is added as a new declaration in the database theory.

6 Translating Queries

Recall that MMT has a general-purpose Query Language called QMT [Rab12], which allows users to find knowledge subject to even complex conditions. We continue by briefly addressing **P3**: query translation; for a complete discussion we refer the interested reader to [Wie17].

In practice, most queries involving virtual theories so far have a shape similar to the one that LMFDB supports: Finding all objects within a single sub-database for which a specific field has a specific value. As an example, consider again the query of finding all Abelian transitive groups. QMT has an MMT-powered surface syntax, which can be used to express this query as:

```
x in (related to ( literal ‘lmfdb:db/transitivegroups?group ) by (object declares))
  | holds x (x commutative x *=* true)
```

The example consists of two parts, first we find all objects declared in the `lmfdb:db/transitivegroups?group` theory (line 1), and then we restrict this set of results to all those for which the `commutative` property is `true` (line 2). Notice that this the example shown here is the formal equivalent of the LMFDB query shown in Sect. 3.2. The key difference is that this query does not require knowing the record structure of LMFDB – apart from knowing the proper sub-db, instead it only relies on knowing the mathematical semantics (commutativity) of the query in question.

Recall that to evaluate a query prior to the introduction of virtual theories, the MMT system loaded the theory graph into main memory and then interleaved incremental flattening and query evaluation operations on the MMT data structures until a result had been produced. But it is infeasible to first load all potentially relevant data into memory, and only then proceed with evaluation. This would require loading a copy of LMFDB into main memory, something that virtual theories were designed to avoid.

The low-level API of LMFDB and similar system provides a new approach for making queries towards virtual theories. First, the MMT query is translated into a system-specific information-retrieval language – in the case of LMFDB this is a MongoDB-based syntax. Next, this translated query is sent to the external API. Upon receiving the results, these are translated back into OMDoc/MMT with the help of already existing functionality in the appropriate virtual theory backend.

This leaves just one problem unsolved – translating queries into the system-specific API. However, it is insufficient to simply translate queries as a whole: One hand a general QMT query may or may not involve a virtual theory, on the other hand, it may also involve several (unrelated) virtual theories. This makes it necessary to filter out queries involving virtual theories, and assign them to a specific backend, and then translate only these parts.

Achieving this automatically is a non-trivial problem. Queries are inductive in nature, and one could attempt to intercept each of the intermediate results. However, this would require a check on each intermediate result to first determine if it comes from a virtual theory or not, and then potentially switching the entire evaluation strategy, leading to a computationally expensive implementation.

Instead of intercepting each result, we extended the Query Language to allows users to annotate sub-queries for evaluation with a specific virtual theory backend. This allows the system to immediately know which parts of a query have to be evaluated in MMT memory, and which have to be translated and sent to an external system. This turns the example above into:

```
use"lmfdb"for {*
  x in (related to ( literal ‘lmfdb:db/transitivegroups?group )
    by (object declares)) | holds x (x commutative x *=* true)
*}
```

Here, we have simply wrapped the entire query with a `use lmfdb` statement, indicating the query should be evaluated using LMFDB.

The encoding of this specific query can be achieved using codecs – in fact we have already seen above in Sect. 3.2 how this is achieved. The query corresponds to the URL http://www.lmfdb.org/api/transitivegroups/groups/?ab=1. Next, the LMFDB API returns a set of JSON objects corresponding to all Abelian transitive groups. These can then be decoded into OMDoc/MMT objects using the procedure described in Sect. 5.2, i.e. for each field we look up the corresponding codec and use it to deconstruct the field, eventually creating an MMT record. Afterwards, these OMDoc/MMT terms can then be passed to the user as a result to the query.

7 Conclusion

We have shown how to extend the Math-in-the-Middle framework for integrating systems to mathematical data bases like the LMFDB. The main idea is to embed knowledge sources as virtual theories, i.e. theories that are not – theoretically or in practice – limited in the number of declarations and allow dynamic loading and processing. For accessing real-world knowledge sources, we have developed the notion of codecs and integrated them into the MitM ontology framework. These codecs (and their MitM types) lift knowledge source access to the MitM level and thus enable object-level interoperability and allow humans (mathematicians) access using the concepts they are familiar with. Finally, we have shown a prototypical query translation facility that allows to delegate some of the processing to the underlying knowledge source and thus avoid thrashing of virtual theories.

Related Work. Most other integration schemes employ a **homogenous approach**, where there is a master system and all data is converted into that system. A paradigmatic example of this is the Wolfram Language [Wik17] and the Wolfram Alpha search engine [Wol], which are based on the Mathematica kernel. This is very flexible for anyone owning a Mathematica license and experienced in the Mathematica language and environment.

The MitM-based approach to interoperability of data sources and systems proposed in this paper is inherently a **heterogeneous approach**: systems and data sources are kept "as is", but their APIs are documented in a machine-actionable way that can be utilized for remote procedure calls, content format mediation, and service discovery. As a consequence, interaction between systems is very flexible. For the data source integration via virtual theories presented in this paper this is important. For instance, we can just make an extension of MMT or Sage which just act as a programmatic interface for e.g. LMFDB.

Future Work. We have discussed the MitM+virtual theories methodology on the elliptic curves sub-base of the LMFDB, which we have fully integrated. We are currently working on additional LMFDB sub-bases. The main problem to be

solved is to elicit the information for the respective schema theories from the LMFDB community. Once that is accomplished, specifying them in the format discussed in this paper and writing the respective codecs is straightforward.

Moreover, we are working on integrating the Online Encyclopedia of Integer Sequences (OEIS [Slo03,OEIS]). Here we have a different problem: the OEIS database is essentially a flat ASCII file with different slots (for initial segments of the sequences, references, comments, and formulae); all minimally marked up ASCII art. In [LK16] we have already (heuristically) flexiformalized OEIS contents in OMDoc/MMT; the next step will be to come up with codecs based on this basis and develop schema theories for OEIS.

Acknowledgements. The authors gratefully acknowledge the fruitful discussions with other participants of work package WP6, in particular John Cremona on the LMFDB and Dennis Müller on early versions of the OMDoc/MMT-based integration. We acknowledge financial support from the OpenDreamKit Horizon 2020 European Research Infrastructures project (#676541).

References

[Cre16] Cremona, J.: The L-functions and modular forms database project. Found. Comput. Math. **16**(6), 1541–1553 (2016). https://doi.org/10.1007/s10208-016-9306-z

[Deh+16] Dehaye, P.-O., et al.: Interoperability in the OpenDreamKit project: the Math-in-the-Middle approach. In: Kohlhase, M., Johansson, M., Miller, B., de Moura, L., Tompa, F. (eds.) CICM 2016. LNCS (LNAI), vol. 9791, pp. 117–131. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42547-4_9. https://github.com/OpenDreamKit/OpenDreamKit/blob/master/WP6/CICM2016/published.pdf

[FGT92] Farmer, W.M., Guttman, J.D., Javier Thayer, F.: Little theories. In: Kapur, D. (ed.) CADE 1992. LNCS, vol. 607, pp. 567–581. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55602-8_192

[Koh+17] Kohlhase, M., et al.: Knowledge-based interoperability for mathematical software systems. In: Blömer, J., Kutsia, T., Simos, D. (eds.) MACIS 2017. LNCS, vol. 10693, pp. 195–210. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-72453-9_14. https://github.com/OpenDreamKit/OpenDreamKit/blob/master/WP6/MACIS17-interop/crc.pdf

[Koh06] Kohlhase, M.: OMDoc – An Open Markup Format for Mathematical Documents [Version 1.2]. LNCS (LNAI), vol. 4180. Springer, Heidelberg (2006). https://doi.org/10.1007/11826095. http://omdoc.org/pubs/omdoc1.2.pdf

[LK16] Luzhnica, E., Kohlhase, M.: Formula semantification and automated relation finding in the On-line Encyclopedia for integer sequences. In: Greuel, G.-M., Koch, T., Paule, P., Sommese, A. (eds.) ICMS 2016. LNCS, vol. 9725, pp. 467–475. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42432-3_60

[Lmf] LMFDB - API. http://www.lmfdb.org/api/. Accessed 17 Sept 2017

[LMFDB] The LMFDB Collaboration: The L-functions and modular forms database. http://www.lmfdb.org. Accessed 01 Feb 2016

[MH] MathHub.info: Active mathematics. http://mathhub.info. Accessed 28 Jan 2014

[MMT] MMT - Language and System for the Uniform Representation of Knowledge. Project web site. https://uniformal.github.io/. Accessed 30 Aug 2016

[ODK] OpenDreamKit Open Digital Research Environment Toolkit for the Advancement of Mathematics. http://opendreamkit.org. Accessed 21 May 2015

[OEIS] OEIS Foundation Inc. (ed.): The On-line Encyclopedia of Integer Sequences. http://oeis.org. Accessed 28 May 2017

[Rab12] Rabe, F.: A query language for formal mathematical libraries. In: Jeuring, J., Campbell, J.A., Carette, J., Dos Reis, G., Sojka, P., Wenzel, M., Sorge, V. (eds.) CICM 2012. LNCS (LNAI), vol. 7362, pp. 143–158. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31374-5_10. arXiv:1204.4685 [cs.LO]

[Rab13] Rabe, F.: The MMT API: a generic MKM system. In: Carette, J., Aspinall, D., Lange, C., Sojka, P., Windsteiger, W. (eds.) CICM 2013. LNCS (LNAI), vol. 7961, pp. 339–343. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39320-4_25

[RK13] Rabe, F., Kohlhase, M.: A scalable module system. Inf. Comput. **230**, 1–54 (2013). http://kwarc.info/frabe/Research/mmt.pdf

[Slo03] Sloane, N.J.A.: The On-line Encyclopedia of integer sequences. Not. AMS **50**(8), 912 (2003)

[Wie17] Wiesing, T.: Enabling cross-system communication using virtual theories and QMT. Master's thesis, Jacobs University Bremen, Bremen, Germany, August 2017. https://github.com/tkw1536/MasterThesis/raw/master/thesis.pdf

[Wik17] Wikipedia: Wolfram language – Wikipedia, the free Encyclopedia (2017). https://en.wikipedia.org/w/index.php?title=Wolfram_Language. Accessed 09 Oct 2017

[Wol] Wolfram-Alpha. http://www.wolframalpha.com. Accessed 05 Jan 2013

[LMFa] The LMFDB Collaboration: The L-functions and modular forms database. http://www.lmfdb.org. Accessed 27 Aug 2016

On Real Roots Counting for Non-radical Parametric Ideals

Ryoya Fukasaku(✉) and Yosuke Sato

Tokyo University of Science, Tokyo, Japan
fukasaku@rs.tus.ac.jp, ysato@rs.kagu.tus.ac.jp

Abstract. An algorithm we have introduced has a great effect on quantifier elimination of a first order formula containing many equalities. When the parametric ideal generated by the underlying equalities is not radical, however, our algorithm tends to produce an unnecessarily complicated formula. In this short paper, we show a result concerning Hermitian quadratic forms. It enables us to improve our algorithm so that we can get a simple formula without any radical computation.

Keywords: Hermitian quadratic form
Comprehensive Gröbner system · Quantifier elimination

1 Introduction

We have introduced an algorithm in [2] as a special type of a Quantifier Elimination (QE) algorithm. It has a great effect on QE of a first order formula containing many equalities. The essential part of the algorithm is to eliminate all existential quantifiers $\exists \bar{X}$ from the following basic first order formula:

$$\phi(\bar{A}) \wedge \exists \bar{X} \, (\bigwedge_{1 \leq i \leq s} f_i(\bar{A}, \bar{X}) = 0 \wedge \bigwedge_{1 \leq i \leq t} h_i(\bar{A}, \bar{X}) > 0) \tag{1}$$

with polynomials $f_1, \ldots, f_s, h_1, \ldots, h_t$ in $\mathbb{Q}[\bar{A}, \bar{X}]$ such that the parametric ideal $I = \langle f_1, \ldots, f_s \rangle$ in $\mathbb{C}[\bar{X}]$ is zero-dimensional for any specialization of the parameters $\bar{A}$ satisfying $\phi(\bar{A})$, where $\phi(\bar{A})$ is a quantifier free formula consisting only of equality $=$ and disequality $\neq$. The algorithm computes a Comprehensive Gröbner System (CGS) of the parametric ideal I, then applies the method of [6] (we call *CGS-QE* method in this paper) which is based on the theory of real roots counting by a Hermitian Quadratic Form (HQF) introduced in [5] with several innovative improvements. The algorithm is further improved by several techniques reported in [3] and implemented in Maple as freeware software [4]. It achieves a good performance for first order formulas containing many equalities as is reported in [1]. When the parametric ideal I is not radical, however, our algorithm tends to produce a unnecessarily complicated formula. Although we may get a simpler formula by computing a CGS of the radical ideal $\sqrt{I}$, such a computation is very heavy in general in a parametric polynomial ring.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 258–263, 2017.
https://doi.org/10.1007/978-3-319-72453-9_18

In this paper, we study the structure of a HQF and show a result namely Theorem 8. It enables us to compute a quantifier free formula equivalent to (1) which is as simple as the one obtained using a CGS of $\sqrt{I}$ without any radical computation. The paper is organized as follows. In Sect. 2, we give a quick review of our CGS-QE algorithm for understanding our result. In Sect. 3, we introduce our main result together with an example which is simple but enough for understanding how we can improve our CGS-QE algorithm.

2 Preliminary

2.1 Multivariate Real Roots Counting

In the rest of the paper, $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$ denote the fields of rational numbers, real numbers and complex numbers respectively. $\bar{X}$ and $\bar{A}$ denote some variables $X_1, \ldots, X_n$ and $A_1, \ldots, A_m$. $T(\bar{X})$ denotes a set of terms in $\bar{X}$. For an ideal $I \subset \mathbb{R}[\bar{X}]$, let $V_{\mathbb{R}}(I) = \{\bar{c} \in \mathbb{R}^n | \forall f \in I \ f(\bar{c}) = 0\}$ and $V_{\mathbb{C}}(I) = \{\bar{c} \in \mathbb{C}^n | \forall f \in I \ f(\bar{c}) = 0\}$. Let I be a zero dimensional ideal in a polynomial ring $\mathbb{R}[\bar{X}]$. Considering the residue class ring $\mathbb{R}[\bar{X}]/I$ as a vector space over $\mathbb{R}$, let $v_1, \ldots, v_q$ be its basis. For an arbitrary $h \in \mathbb{R}[\bar{X}]/I$ and each i, j $(1 \leq i, j \leq q)$ we define a linear map $\theta_{h,i,j}$ from $\mathbb{R}[\bar{X}]/I$ to $\mathbb{R}[\bar{X}]/I$ by $\theta_{h,i,j}(f) = hv_iv_jf$ for $f \in \mathbb{R}[\bar{X}]/I$. Let $q_{h,i,j}$ be the trace of $\theta_{h,i,j}$ and M_h^I be a real symmetric matrix such that its (i, j)-th component is given by $q_{h,i,j}$. Regarding a real symmetric matrix as a quadratic form, M_h^I is called, a *Hermitian Quadratic Form (HQF)*. The characteristic polynomial of M_h^I is denoted by $\chi_h^I(x)$. The dimension of $\mathbb{R}[\bar{X}]/I$ is denoted by $\dim(\mathbb{R}[\bar{X}]/I)$. For a polynomial $f(x) \in \mathbb{R}[x]$, the signature of $f(x)$, denoted $\mathrm{sign}(f(x))$, is an integer which is equal to 'the number of positive real roots of $f(x) = 0$' − 'the number of negative real roots of $f(x) = 0$', that is, $\mathrm{sign}(f(x)) = \#(\{c \in \mathbb{R} | f(c) = 0, c > 0\}) - \#(\{c \in \mathbb{R} | f(c) = 0, c < 0\})$. The signature of M_h^I, denoted $\mathrm{sign}(M_h^I)$, is defined as the signature of $\chi_h^I(x)$. The real root counting theorem introduced in [5] is the following assertion.

Theorem 1. $\mathrm{sign}(M_h^I) = \#(\{\bar{x} \in V_{\mathbb{R}}(I) | h(\bar{x}) > 0\}) - \#(\{\bar{x} \in V_{\mathbb{R}}(I) | h(\bar{x}) < 0\})$.

2.2 Comprehensive Gröbner System

Definition 2. *For a subset $\mathcal{S}$ of $\mathbb{C}^m$, a finite set $\{\mathcal{S}_1, \ldots, \mathcal{S}_r\}$ of subsets of $\mathbb{C}^m$ which satisfies $\cup_{i=1}^r \mathcal{S}_i = \mathcal{S}$ and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset (i \neq j)$ is called a* partition *of $\mathcal{S}$. Each $\mathcal{S}_i$ is called a* segment.

Definition 3. *Let $\succ$ be an admissible term order on $T(\bar{X})$. For a polynomial $f \in \mathbb{C}[\bar{A}, \bar{X}]$, regarding f as a member of a polynomial ring $\mathbb{C}[\bar{A}][\bar{X}]$ over a coefficient ring $\mathbb{C}[\bar{A}]$, its leading term and coefficient are denoted by $LT_{\succ}(f)$ and $LC_{\succ}(f)$ respectively. For a finite set $F \subset \mathbb{Q}[\bar{A}, \bar{X}]$ and a subset $\mathcal{S}$ of $\mathbb{C}^m$, a finite set of pairs $\mathcal{G} = \{(\mathcal{S}_1, G_1), \ldots, (\mathcal{S}_r, G_r)\}$ with finite sets G_i of $\mathbb{Q}[\bar{A}, \bar{X}]$ for each i satisfying the following properties 1, 2, 3 is called a* (minimal) comprehensive Gröbner system (CGS) *of $\langle F \rangle$ on $\mathcal{S}$ with parameters $\bar{A}$ w.r.t. the term order $\succ$.*

1. $\{\mathcal{S}_1, \ldots, \mathcal{S}_r\}$ *is a partition of* $\mathcal{S}$.
2. *For each* i *and any* $\bar{a} \in \mathcal{S}_i$, $G_i(\bar{a})$ *is a (minimal) Gröbner basis of* $\langle F(\bar{a})\rangle \subset \mathbb{C}[\bar{X}]$ *w.r.t.* $\succ$, *where* $G_i(\bar{a}) = \{g(\bar{a}, \bar{X}) | g(\bar{A}, \bar{X}) \in G_i\}$ *and* $F(\bar{a}) = \{f(\bar{a}, \bar{X}) | f(\bar{A}, \bar{X}) \in F\}$.
3. *For each* i, $\mathrm{LC}_{\succ}(g)(\bar{a}) \neq 0$ *for every* $g \in G_i$ *and* $\bar{a} \in \mathcal{S}_i$.

Remark 4. *The set of leading terms of* $G_i(\bar{a})$ *is invariant for each* $\bar{a} \in \mathcal{S}_i$, *hence the dimension of the ideal* $\langle G_i(\bar{a})\rangle$ *is also invariant. A minimal CGS is desirable for their computation. When the ideal* $\langle G_i(\bar{a})\rangle$ *is zero-dimensional for* $\bar{a} \in \mathcal{S}_i$, *using* $(\mathcal{S}_i, G_i)$ *we can also compute a uniform representation of the HQF* M_h^I *on* $\mathcal{S}_i \cap \mathbb{R}^m$ *for any polynomial* $h \in \mathbb{Q}[\bar{A}, \bar{X}]$. *More precisely, each element is represented by a rational function* $p(\bar{A})/q(\bar{A})$ *with* $p(\bar{A}), q(\bar{A}) \in \mathbb{Q}[\bar{A}]$ *such that* $q(\bar{a}) \neq 0$ *is guaranteed for any* $\bar{a} \in \mathcal{S}_i \cap \mathbb{R}^m$.

2.3 CGS-QE Algorithm

The following result is the most important contribution of our paper [2] for the elimination of the quantifiers $\exists \bar{X}$ from the basic first order formula (1) given in Sect. 1.

Theorem 5. *Let* $\mathcal{S} = \{\bar{a} \in \mathbb{C}^m | \phi(\bar{a})\}$ *and* $\mathcal{G} = \{(\mathcal{S}_1, G_1), \ldots, (\mathcal{S}_r, G_r)\}$ *be a minimal CGS of the parametric saturation ideal* $I : h^\infty$ *on* $\mathcal{S}$ *with parameters* $\bar{A}$ *w.r.t. an arbitrary term order, where* $I = \langle f_1, \ldots, f_s\rangle$ *and* $h = \prod_{1 \leq i \leq t} h_i$. *For each* i *and any* $\bar{a} \in \mathcal{S}_i \cap \mathbb{R}^m$, *the followings are equivalent:*

1. $\exists \bar{X}\ (\bigwedge_{1 \leq i \leq s} f_i(\bar{a}, \bar{X}) = 0 \wedge \bigwedge_{1 \leq i \leq t} h_i(\bar{a}, \bar{X}) > 0)$.
2. $\sum_{(e_1, \ldots, e_t) \in \{0,1\}^t} \mathrm{sign}(M^{\langle G_i(\bar{a})\rangle}_{h_1^{e_1} \cdots h_t^{e_t}(\bar{a})}) > 0$.

By Remark 4, for each i we have a uniform representation of $M^{\langle G_i(\bar{a})\rangle}_{h_1^{e_1} \cdots h_t^{e_t}(\bar{a})}$ for $\bar{a} \in \mathcal{S}_i \cap \mathbb{R}^m$. Using it together with Descartes' rule of signs, we can construct a quantifier free first order formula $\psi_i(\bar{A})$ such that $\psi_i(\bar{a})$ is equivalent to the property 2 of Theorem 5 for each $\bar{a} \in \mathcal{S}_i \cap \mathbb{R}^m$, then we have a quantifier free first order formula $\phi(\bar{A}) \wedge (\bigvee_{1 \leq i \leq r} \psi_i(\bar{A}))$ equivalent to (1).

An essential and important difference between our CGS-QE algorithm of [2] and the original CGS-QE algorithm of [6] is that our algorithm computes a CGS of the saturation ideal $I : h^\infty$ whereas the original computes a CGS of I and use the relation $\sum_{(e_1, \ldots, e_t) \in \{1,2\}^t} \mathrm{sign}(M^{\langle G_i(\bar{a})\rangle}_{h_1^{e_1} \cdots h_t^{e_t}(\bar{a})}) > 0$ which is also equivalent to the property 1 of Theorem 5. When $I : h^\infty \neq I$, we have $\dim(\mathbb{R}[\bar{X}]/I) > \dim(\mathbb{R}[\bar{X}]/I : h^\infty)$ and the size of $M^{\langle G_i(\bar{a})\rangle}_{h_1^{e_1} \cdots h_t^{e_t}(\bar{a})}$ is smaller in our algorithm, which enables us to have a simpler representation formula of $\psi_i(\bar{A})$. Even when $I : h^\infty = I$, we also have its simpler representation since the polynomial $h_1^{e_1} \cdots h_t^{e_t}$ becomes more complicated if we allow $e_1, \ldots, e_t$ to be 2.

3 New Multivariate Real Roots Counting

As is mentioned at the end of the last section, the size of the HQF $M^{\langle G_i(\bar{a})\rangle}_{h_1^{e_1}\cdots h_t^{e_t}(\bar{a})}$ effects the simplicity of the representation formula of $\psi_i(\bar{A})$. Note that we can replace $M^{\langle G_i(\bar{a})\rangle}_{h_1^{e_1}\cdots h_t^{e_t}(\bar{a})}$ with $M^{\sqrt{\langle G_i(\bar{a})\rangle}}_{h_1^{e_1}\cdots h_t^{e_t}(\bar{a})}$ in Theorem 5. If $\langle G_i(\bar{a})\rangle$ is not a radical ideal, $\dim(\mathbb{R}[\bar{X}]/\langle G_i(\bar{a})\rangle) > \dim(\mathbb{R}[\bar{X}]/\sqrt{\langle G_i(\bar{a})\rangle})$ and we may have a simpler representation formula of $\psi_i(\bar{A})$ using a CGS of the radical ideal $\sqrt{I:h^\infty}$.

Example 6. *Consider the following simple example in a form of the basic first order formula:* $A \neq 0 \wedge \exists X((X-A)^2 = 0 \wedge X > 0)$. $\phi(A)$ *is* $A \neq 0$, *the parametric ideal* I *is* $\langle (X-A)^2\rangle$ *and* $h = X$. *A minimal CGS* $\mathcal{G}$ *of the parametric saturation ideal* $I : h^\infty$ *on* $\mathcal{S} = \{a \in \mathbb{C} | a \neq 0\}$ *has the form* $\mathcal{G} = \{(\mathcal{S}, \{(X-A)^2\})\}$, *whereas a minimal CGS* $\mathcal{G}'$ *of the radical ideal* $\sqrt{I:h^\infty}$ *on* $\mathcal{S}$ *has the form* $\mathcal{G}' = \{(\mathcal{S}, \{X-A\})\}$. *Let* $G = \{(X-A)^2\}$ *and* $G' = \{X-A\}$. *We have the following uniform representations of the HQFs on* $\mathcal{S} \cap \mathbb{R}$:

$$M_1^{\langle G\rangle} = \begin{pmatrix} 2 & 2A \\ 2A & 2A^2 \end{pmatrix}, \quad M_X^{\langle G\rangle} = \begin{pmatrix} 2A & 2A^2 \\ 2A^2 & 2A^3 \end{pmatrix}, \quad M_1^{\langle G'\rangle} = \begin{pmatrix} 1 \end{pmatrix}, \quad M_X^{\langle G'\rangle} = \begin{pmatrix} A \end{pmatrix}.$$

Applying Descartes' rule of signs and the simplification technique introduced in the Sect. 3 of [3] *to the characteristic polynomials of* $M_1^{\langle G\rangle}$ *and* $M_X^{\langle G\rangle}$, *(although we do not need them for this simple example), we have an equivalent quantifier free formula:*

$$A \neq 0 \wedge A^2 + 1 > 0 \wedge A^3 + A > 0.$$

On the other hand, if we use the characteristic polynomials of $M_1^{\langle G'\rangle}$ *and* $M_X^{\langle G'\rangle}$, *we have a much simpler equivalent quantifier free formula:*

$$A \neq 0 \wedge 1 > 0 \wedge A > 0.$$

3.1 Main Result

Though a CGS of the radical ideal $\sqrt{I:h^\infty}$ may reduce the size of HQFs, a radical computation is generally very heavy for a parametric polynomial ring. In this section, we show a new result Theorem 8. It brings us a new CGS-QE method which does not use any radical computation but produces a quantifier free formula as simple as the one obtained using a CGS of the radical ideal.

Notation 7. *For a* $q \times q$ *square matrix* M *and* $1 \leq b_1 < \ldots < b_k \leq q$, $M(b_1, \ldots, b_k)$ *denotes a* $k \times k$ *square matrix such that its* (i,j)*-th component is the* (b_i, b_j)*-th component of* M *for each* $i, j (1 \leq i, j \leq k)$.

We have the following property similar to Theorem 1.

Theorem 8. *Let* I *be a zero-dimensional ideal of* $\mathbb{R}[\bar{X}]$ *such that* $\dim(R[\bar{X}]/I) = q$ *and* $\mathrm{rank}(M_1^I) = k$, *note that* M_1^I *is a* $q \times q$ *matrix, hence* $k \leq q$. *Then there exists a* $k-$*tuple* $(b_1, \ldots, b_k)$ *of integers such that* $1 \leq b_1 < \ldots < b_k \leq q$ *and*

$\det(M_1^I(b_1,\ldots,b_k)) \neq 0$. *For any such a* $k-$*tuple, we have the following equation for every polynomial* $h \in \mathbb{R}[\bar{X}]$*:*

$$\mathrm{sign}(N_h^I) = \#(\{\bar{x} \in V_{\mathbb{R}}(I) | h(\bar{x}) > 0\}) - \#(\{\bar{x} \in V_{\mathbb{R}}(I) | h(\bar{x}) < 0\}),$$

where N_h^I *denote a* $k \times k$ *real symmetric matrix* $M_h^I(b_1,\ldots,b_k)$.

By this theorem, we can replace $M^{\langle G_i(\bar{a})\rangle}_{h_1^{e_1}\cdots h_t^{e_t}(\bar{a})}$ with $N^{\langle G_i(\bar{a})\rangle}_{h_1^{e_1}\cdots h_t^{e_t}(\bar{a})}$ in Theorem 5. Since $\dim(\mathbb{R}[\bar{X}]/\sqrt{\langle G_i(\bar{a})\rangle}\ ) = \mathrm{rank}(M_1^{\langle G_i(\bar{a})\rangle})$ by the theory of roots counting, $N^{\langle G_i(\bar{a})\rangle}_{h_1^{e_1}\cdots h_t^{e_t}(\bar{a})}$ and $M^{\sqrt{\langle G_i(\bar{a})\rangle}}_{h_1^{e_1}\cdots h_t^{e_t}(\bar{a})}$ have a same size and we can obtain a simple formula.

Example 9. *For the HQF* $M_1^{\langle G\rangle}$ *in the previous example,* $q = 2$ *and* $k = 1$*. We may have* $b_1 = 1$ *or* $b_1 = 2$*. For* $b_1 = 1$*, we have* $N_1^{\langle G\rangle} = (2)$ *and* $N_X^{\langle G\rangle} = (2A)$ *which produces the same formula* $A \neq 0 \wedge 1 > 0 \wedge A > 0$ *as the formula obtained using the radical ideal. On the other hand, for* $b_1 = 2$*, we have* $N_1^{\langle G\rangle} = (2A)$ *and* $N_X^{\langle G\rangle} = (2A^3)$ *which produces the formula* $A \neq 0 \wedge A^2 > 0 \wedge A^3 > 0$.

4 Conclusion and Remarks

In Example 9, the obtained formula for $b_1 = 2$ does not look much simpler than the one obtained using $M_1^{\langle G\rangle}$ and $M_X^{\langle G\rangle}$. Though the formula obtained using any k-tuple $(b_1,\ldots,b_k)$ is generally simple for a more complicated non-radical ideal I, the choice of k-tuple makes a strong effect on its simplicity. For the choice of a k-tuple we also have obtained the following criterion. Let $\succ$ be an admissible term order of $T(\bar{X})$ and $\{v_1,\ldots,v_q\} = \{v \in T(\bar{X}) | v \notin LT(I)\}$. We can choose $(b_1,\ldots,b_k)$ so that each v_{b_i} is not dividable by v_j for any $j \in \{1,\ldots,q\} \setminus \{b_1,\ldots,b_k\}$. Such a k-tuple produces a simple formula. Note that in the previous example, $b_1 = 1$ satisfies this criterion but $b_1 = 2$ does not.

References

1. http://www.rs.tus.ac.jp/fukasaku/software/CGSQE-20160509/
2. Fukasaku, R., Iwane, H., Sato, Y: Real quantifier elimination by computation of comprehensive Gröbner systems. In: Proceedings of International Symposium on Symbolic and Algebraic Computation, pp. 173–180, ACM-Press (2015)
3. Fukasaku, R., Iwane, H., Sato, Y.: On the implementation of CGS real QE. In: Greuel, G.-M., Koch, T., Paule, P., Sommese, A. (eds.) ICMS 2016. LNCS, vol. 9725, pp. 165–172. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42432-3_21
4. Fukasaku, R., Iwane, H., Sato, Y.: CGSQE/SyNRAC: a real quantifier elimination package based on the computation of comprehensive Gröbner systems. ACM Comm. Comput. Algebra **50**(3), 101–104 (2016)

5. Pedersen, P., Roy, M.-F., Szpirglas, A.: Counting real zeros in the multivariate case. In: Eyssette, F., Galligo, A. (eds.) Computational Algebraic Geometry. PM, vol. 109, pp. 203–224. Birkhäuser Boston, Boston (1993). https://doi.org/10.1007/978-1-4612-2752-6_15
6. Weispfenning, V.: A new approach to quantifier elimination for real algebra. In: Caviness, B.F., Johnson, J.R. (eds.) Quantifier Elimination and Cylindrical Algebraic Decomposition, pp. 376–392. Springer, Vienna (1998). https://doi.org/10.1007/978-3-7091-9459-1_20

On the Bit-Size of Non-radical Triangular Sets

Xavier Dahan(✉)

Faculty of General Educational Research, Ochanomizu university, Tokyo, Japan
dahan.xavier@ocha.ac.jp

Abstract. We present upper bounds on the bit-size of coefficients of non-radical purely lexicographical Gröbner bases (triangular sets) in dimension zero. This extends a previous work [4], constrained to radical triangular sets; it follows the same technical steps, based on interpolation. However, key notion of height of varieties is not available for points with multiplicities; therefore the bounds obtained are thus less universal and depend on some input data. We also introduce a related family of non-monic polynomials that have smaller coefficients, and smaller bounds. It is not obvious to compute them from the initial triangular set though.

1 Introduction

Triangular sets are the core objects of the triangular decomposition methods to solve polynomial systems [1,2,6,7,11]. Algorithms in this realm are somewhat based on the generalization of algorithms for *univariate* polynomials to multivariate ones, yielding some splittings viz. "decomposition". The outputs are most of the time *regular chains* (a.k.a regular sets). In dimension zero they can be made particularly simple: they form a reduced lexicographic Gröbner basis $(t_1(x_1), t_2(x_1, x_2), \ldots, t_n(x_1, \ldots, x_n))$. We will refer to such a family as a *triangular set* in this article.

To solve polynomial systems, it is enough to represent the radical ideal generated by the input polynomials, thereby most previous works focus on radical triangular sets. However, triangular sets have the ability to represent some non-radical ideals (called thereafter triangular ideals); Moreover the radical of the ideal generated by a triangular set is not necessarily triangular, requiring extra work to decompose it into triangular sets. If we compare with the Rational Univariate Representation (RUR, see [10]), only the multiplicity (which is just a number) of a root is given. Therefore beyond the theoretical interest, it is worth studying non-radical triangular sets.

In this article we unveil the *structure* and prove upper bounds on the bit-size of coefficients of such triangular sets. This is an attempt of generalization from radical to non-radical triangular sets, of the results given in [4]. Let us recall briefly the strategy of this paper, since we will follow it.

Step (1) Given the solution points, some interpolation formulas are proved to reconstruct the triangular set from the points.

Work supported by the JSPS grant Wakate B No. 50567518.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 264–269, 2017.
https://doi.org/10.1007/978-3-319-72453-9_19

Step (2) These formula allows to control the growth of coefficients in function of that of the points.

Step (3) A tool from Diophantine geometry called *height of variety* defined through a *Chow form* is introduced. It measures somewhat the arithmetic complexity of the variety, and is endowed of an arithmetic analogue of the Bézout theorem (degree of intersection). This "Arithmetic Bézout theorem" provides upper bounds in function of *any* input polynomial system.

Step (4) A simple modification of the interpolation formulas, called *barycentric* form of Lagrange interpolation defines a family of non-monic polynomials which have smaller coefficients.

We present extensions of Steps (1)–(2), and partially (4), to non-radical triangular sets. As for Step (3) the tool (height of variety) is not available for multiple points. While interpolation in [4] is multivariate Lagrange, it is here relevant of multivariate Hermite. The input data are not points but primary ideals assumed to be given by a triangular set (see (1)). The related family of non-monic polynomials mentioned in Step (4) and denoted N_ℓ are defined in Theorem 1. In comparison with [4], they seem not easy to compute from the triangular set T (see [12]) for an attempt in two variables).

Related work. For the bit-size bounds, a selection of related work concerned with the RUR, triangular sets, and lexicographic Gröbner bases is [3,4,8,9]. The bounds presented are the first ones dealing with non-radical systems having a general type of singularities. Comparatively, a RUR can represent multiplicities (recall that this is just a number) but not a full singularity type.

Notation. k will denote any field $\mathbb{Q} \subseteq k \subsetneq \overline{\mathbb{Q}}$. A polynomial ring over k, in n variables $x_1, \ldots, x_n$ implicitly ordered such as $x_1 \prec x_2 \prec \cdots \prec x_n$, a triangular set $T = (T_1(x_1), \ldots, T_n(x_1, \ldots, x_n))$ with $\deg_{x_j}(T_j) = d_j$. Its set of zeros in $\overline{k}^n$ is denoted V. For a subset $\mathcal{S}$ of an arbitrary Cartesian product E^m, $\mathcal{S}_{\leq \ell}$ will denote the projection of $\mathcal{S}$ on the first ℓ coordinates.

2 Interpolation Formula

Input data. In the radical case, we want to interpolate points. Here, the raw input data are primary ideals associated to each solution point in $\overline{k}^n$. Thanks to Theorem 2.4 of [5] the primary ideals of a triangular set are triangular: the lexicographic Gröbner basis of these primary ideals are triangular sets. Still over $\overline{k}$, a prime ideal associated with $\langle T \rangle$ is of the form $\langle x_1 - \alpha_1, \ldots, x_n - \alpha_n \rangle$, for a solution point $(\alpha_1, \ldots, \alpha_n)$. The corresponding primary ideal $\langle t^{(\alpha)} \rangle$ has the following shape (Proposition 2.2 of [5]): First $t_1^{(\alpha)}(x_1) = (x_1 - \alpha_1)^{\delta_1(\alpha)}$ and in general $t_n^{(\alpha)}(x_1, \ldots, x_n)$ is equal to:

$$(x_n - \alpha_n)^{\delta_n(\alpha)} + \sum_{i_1=0}^{\delta_1(\alpha_1)-1} \cdots \sum_{i_{n-1}=0}^{\delta_{n-1}(\alpha)-1} \sum_{i_n=0}^{\delta_n(\alpha)-1} c_\alpha[i_1, \ldots, i_n] \prod_{j=1}^{n} (x_j - \alpha_j)^{i_j} \quad (1)$$

where:

(i) For $1 \leq u \leq n$, $\deg_{x_u}(c[i_1, \ldots, i_n]) < \delta_u(\alpha)$,
(ii) $c[0, \ldots, 0, i_\ell] = 0$ for all $i_\ell < \delta_\ell(\alpha)$ and for $\ell = 2, \ldots, n$.
(iii) Note that Taylor expansion gives: $c[i_1, \ldots, i_n] = \frac{1}{i_1! \cdots i_n!} \frac{\partial^{i_1 + \cdots + i_n} t_n}{\partial x_1^{i_1} \cdots \partial x_n^{i_n}} (\alpha_1, \ldots, \alpha_n)$.

We denote by $T_{\ell+1}[\alpha]$ the polynomial $T_{\ell+1} \bmod \langle t_{\leq \ell}^{(\alpha)} \rangle$. Theorem 3.1 of [5], the ring $\bar{k}[x_1, \ldots, x_n]/\langle t_{\leq \ell}^{(\alpha)} \rangle$ is *Henselian*. Hence $T_{\ell+1}[\alpha]$ admits a unique factorization as follows:

$$T_{\ell+1}[\alpha] \equiv \prod_{\beta \in V_{\leq \ell+1}} t_{\ell+1}^{(\beta)} \bmod \langle t_{\leq \ell}^{(\alpha)} \rangle, \text{where } (\beta_1, \ldots, \beta_\ell) = (\alpha_1, \ldots, \alpha_\ell). \tag{2}$$

and $t_{\ell+1}^{(\beta)} = (x_{\ell+1} - \beta_{\ell+1})^{\delta_{\ell+1}(\beta)} + \sum_{i_1, \cdots, i_\ell, r} c_\beta[i_1, \ldots, i_\ell, r](x_{\ell+1} - \beta_{\ell+1})^r \cdot \prod_{j=1}^{\ell}(x_j - \alpha_j)^{i_j}$ for some $c_\beta[i_1, \ldots, i_\ell, r] \in \bar{k}$. This key result allows to prove Proposition 1.

Notation. A sequence $\alpha^1 \in V_{\leq 1}, \alpha^2 \in V_{\leq 2}, \ldots, \alpha^\ell \in V_{\leq \ell}$ will *not* denote "j-th power of α", but points $\alpha^j = (\alpha_1^j, \ldots, \alpha_j^j)$ with the additional convention that $(\alpha_1^j, \ldots, \alpha_j^j) = (\alpha_1^{j+1}, \ldots, \alpha_j^{j+1})$. We say that α^{j+1} *extends* α^j.

Proposition 1. *Let $\gamma \in V_{\leq \ell+1}$ be a root that extends $\alpha = (\alpha_1, \ldots, \alpha_\ell) \in V_{\leq \ell}$.*

1. $e_{\ell+1}(\gamma) \equiv \frac{T_{\ell+1}[\alpha]}{t_{\ell+1}^{(\gamma)}} \bmod \langle t_{\leq \ell}^{(\alpha)} \rangle$ *is a polynomial in* $(\bar{k}[x_1, \ldots, x_\ell]/\langle t_{\leq \ell}^{(\alpha)} \rangle)[x_{\ell+1}]$
2. *Orthogonality: Given $\beta \neq \gamma \in V_{\leq \ell+1}$:*
$$e_{\ell+1}(\beta) \cdot e_{\ell+1}(\gamma) = 0 \text{ in } A_\alpha := \bar{k}[x_1, \ldots, x_{\ell+1}]/\langle t_1^{(\alpha)}, \ldots, t_\ell^{(\alpha)}, T_{\ell+1}[\alpha] \rangle.$$
3. *There are polynomials $u_{\ell+1}(\gamma)$ and v such that* $u_{\ell+1}(\gamma)e_{\ell+1}(\gamma) + v\, t_{\ell+1}^{(\gamma)} \equiv 1 \bmod \langle t_{\leq \ell}^{(\alpha)} \rangle$, *with* $\deg_{x_{\ell+1}}(u_{\ell+1}(\gamma)) < \deg_{x_{\ell+1}}(t_{\ell+1}^{(\gamma)})$.
4. $e_{\ell+1}(\beta) \equiv 0 \bmod \langle t_{\leq \ell+1}^{(\beta')} \rangle$ *if* $\beta' \neq \beta$, *and* $\widetilde{e_{\ell+1}}(\gamma) \equiv 1 \bmod \langle t_{\leq \ell+1}^{(\gamma)} \rangle$. *Denote* $\widetilde{e_{\ell+1}}(\gamma) \equiv u_{\ell+1}(\gamma)e_{\ell+1}(\gamma) \bmod \langle t_{\leq \ell}^{(\alpha)} \rangle$. *The family* $\{\widetilde{e_{\ell+1}}(\gamma)\}_\gamma$ *is a complete family of orthogonal idempotents of the algebra* A_α.

Have in mind that Lagrange and Hermite interpolation use idempotents.

Theorem 1. *Write $t_{\leq \ell}^{(\alpha)} = (t_1^{(\alpha)}, \ldots, t_\ell^{(\alpha)})$ the triangular sets defining the primary ideal of associated prime $\langle x_1 - \alpha_1, \ldots, x_\ell - \alpha_\ell \rangle$.*

$$T_{\ell+1} \equiv \sum_{\alpha^1 \in V_{\leq 1}} \sum_{\alpha^2 \in V_{\leq 2}} \cdots \sum_{\alpha^\ell \in V_{\leq \ell}} \widetilde{e_1}(\alpha^1) \cdots \widetilde{e_\ell}(\alpha^\ell) \cdot T_{\ell+1}[\alpha^\ell] \bmod \langle t_{\leq \ell}^{(\alpha)} \rangle. \tag{3}$$

where it is assumed that $\alpha^{j+1} \in V_{\leq j+1}$ extends $\alpha^j \in V_{\leq j}$. We define $N_{\ell+1}$ by the same *formula, using polynomials $e_i(\alpha^i)$ instead of $\widetilde{e_i}(\alpha^i)$.*

Since the family of idempotents used to define $T_{\ell+1}$ is complete, $T_{\ell+1}$ is monic. This is not the case for $N_{\ell+1}$. Therefore they cannot be used to perform *reduction* through a division, but their interest lies in their small coefficients, and in that they generate the same ideal as $\langle T_1, \ldots, T_\ell \rangle$ hence encode the same information; In the radical case they were used in conjunction of modular methods.

A natural question is whether we can compute the polynomials N_ℓ's from the T_ℓ's. The answer is not trivial and not addressed in these pages. However it is not unreasonable to expect an almost linear complexity algorithm to compute them, as shown for the case of two variables in [12]. It boils down to compute the polynomial denoted $F_{\ell+1}$ hereunder

Proposition 2. *We have* $F_{\ell+1}T_{\ell+1} \equiv N_{\ell+1} \bmod \langle T_1, \ldots, T_\ell \rangle$, *with:*

$$F_{\ell+1} := \sum_{\alpha^1 \in V_{\leq 1}} \sum_{\alpha^2 \in V_{\leq 2}} \cdots \sum_{\alpha^\ell \in V_{\leq \ell}} e_1(\alpha^1) \cdots e_\ell(\alpha^\ell)$$

and where the same convention on $\alpha^1, \alpha^2, \ldots$ *as in Theorem 1 is adopted.*

In the radical case, it is easy to show that $F_\ell = \frac{\partial T_1}{\partial x_1} \cdots \frac{\partial T_\ell}{\partial x_\ell}$.

3 Bit-Size Consideration

Preliminary. This last section states and comments on some upper-bounds on the bit-size of coefficients in $\mathbb{Q}$ appearing in the polynomials T and N. The difficulties compared to the radical case are first, that the interpolation formulas are more complicated to handle, and second, that there is no notion of Chow form, yet of notion height of varieties, in our non-radical context; Whereas it was a key tool in [4] to obtain *intrinsic* bit-size bounds (see Step (3) in Introduction). The upper bounds that we can obtain are less universal: we assume that the input primary ideals are given in triangular form, which we have written $t^{(\alpha)}$, while universal bounds would not need this assumption (another input for the primary ideals may well have smaller coefficients, hence yield better bounds). However bounds (6) give a reasonable indication on the bit-size, and are also of interest to understand the growth of coefficients in multivariate (barycentric) Hermite interpolation.

Statement. We use the formalism of *height* of polynomials classical in Diophantine approximation theory. The notation $h(f)$ denotes the height of the polynomial f, and can be thought as the max bit-size of its coefficients. Recall that the input "raw" data are the triangular sets $\{t^{(\alpha)}, \alpha \in V\}$ generating the primary ideals of $\langle T \rangle$. With the notations of (1), this includes the exponents $\delta_i(\alpha)$ and coefficients $c_\alpha[i_1, \ldots, i_\ell]$. We define:

$$H_\ell(\beta^\ell) := \max_{i_1,\ldots,i_\ell} h(c_\beta[i_1, \ldots, i_\ell]) + i_1 h(\beta_1) + \cdots + i_\ell h(\beta_\ell), \tag{4}$$

and $\mathsf{L}_\ell(T) := \max_{\alpha \in V_{\leq \ell}} (\sum_{i=1}^{\ell} H_i(\beta^i))$. Denote by $\mu_\ell(\beta^\ell) := \delta_1(\beta_1) \cdots \delta_\ell(\beta_\ell)$ the local multiplicity at β^ℓ, and finally:

$$H_\ell(T) := \sum_{\beta^\ell \in V_{\leq \ell}} H_\ell(\beta^\ell), \quad \mu_\ell(T) := \max_{\beta^\ell \in V_{\leq \ell}} \mu_\ell(\beta^\ell), \quad D_\ell := \sum_{i \leq \ell} d_i \tag{5}$$

The bit-sizes of any coefficient of $N_{\ell+1}$, or $T_{\ell+1}$, are lower than quantities whose dominating terms are respectively:

$$H_{\ell+1}(T)+\widetilde{O}(\mathsf{L}_\ell(T)\cdot D_{\ell+1}\cdot\mu_\ell(T)), \qquad \ell D_{\ell+1}H_{\ell+1}(T)+\widetilde{O}(\ell\cdot\mathsf{L}_\ell(T)\cdot D_{\ell+1}^2\cdot\mu_\ell(T)) \tag{6}$$

Rationale. $\widetilde{O}(\,.\,)$ is a big-Oh notation that hides any additional logarithmic factors. The quantity $H_\ell(\beta^\ell)$ of (4) should be thought as a generalization to primary components of the height of a projective point: coefficients and exponents in the Taylor expansion (1) are simply "naturally" taken into account. If the point is simple, then this quantity coincides with the "traditional" height of a point. In addition, the quantity $H_\ell(T)$ of (5) reflects a certain additivity of the "height" under distinct primary components; this also generalizes the additivity of the height of varieties under disjoint union. Therefore, the quantities involved are "natural" extension to those more traditional used in the case of simple points. The occurrence of quantities like $\mathsf{L}_\ell(T)$ or the multiplicity at a point $\mu_\ell(\beta^\ell)$ is due to technical details inherent to the presence of multiplicities

If T generates a *radical* ideal, then $\mu_\ell(\beta^\ell)=1$, $H_\ell(\beta^\ell)=h(\beta^\ell)$; moreover $H_\ell(T)\approx h(V_{\leq\ell})$, $D_\ell\leq\deg(V_{\leq\ell})$ which are respectively the *height* of the variety $V_{\leq\ell}$ and its degree, and if we discard the values $\mathsf{L}_\ell(T)$ then the bounds in (6) become roughly $h(V_\ell)+\widetilde{O}(\deg(V))$ for $N_{\ell+1}$, and $\deg(V_\ell)h(V_\ell)+\widetilde{O}(\deg(V_{\leq\ell})^2)$ for $T_{\ell+1}$. These bounds are similar to the ones obtained in [4]. This shows that the bounds (6) "faithfully" extend the ones of the radical case. The difference between the bounds for $N_{\ell+1}$ and for $T_{\ell+1}$ is roughly the factor D_ℓ, and this ratio is comparable to that of [4].

Finally, experiments not reported here (but see Table 1 in [12] for some data in two variables) show that the size of polynomials T_ℓ can be dramatically larger than N_ℓ's.

References

1. Aubry, P., Lazard, D., Moreno Maza, M.: On the theories of triangular sets. J. Symb. Comput. **28**(1–2), 45–124 (1999)
2. Chen, C., Moreno-Maza, M.: Algorithms for computing triangular decomposition of polynomial systems. J. Symb. Comput. **47**(6), 610–642 (2012)
3. Dahan, X.: Size of coefficients of lexicographical Gröbner baes. In: ISSAC 2009, pp. 117–126. ACM (2009)
4. Dahan, X., Schost, É.: Sharp estimates for triangular sets. In: ISSAC 2004, pp. 103–110. ACM Press (2004)
5. Dahan, X.: GCD modulo a primary triangular set of dimension zero. In: Proceedings of ISSAC 2017, pp. 109–116. ACM, New York (2017)
6. Moreno Maza, M., Lemaire, F., Xie, Y.: The `RegularChains` library (2005)
7. Hubert, E.: Notes on triangular sets and triangulation-decomposition algorithms I: polynomial systems. In: Winkler, F., Langer, U. (eds.) SNSC 2001. LNCS, vol. 2630, pp. 1–39. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45084-X_1
8. Mantzaflaris, A., Schost, É., Tsigaridas, E.: Sparse rational univariate representation. In: ISSAC 2017, p. 8 (2017)

9. Mehrabi, E., Schost, É.: A softly optimal monte carlo algorithm for solving bivariate polynomial systems over the integers. J. Complex. **34**, 78–128 (2016)
10. Rouillier, F.: Solving zero-dimensional systems through the rational univariate representation. Appl. Algebra Eng. Commun. Comput. **9**(5), 433–461 (1999)
11. Wang, D.: Elimination Methods. Texts and Monographs in Symbolic Computation. Springer, Vienna (2012). https://doi.org/10.1007/978-3-7091-6202-6
12. Yamashita, T., Dahan, X.: Bit-size reduction of triangular sets in two and three variables. In: SCSS 2016. EPiC Series in Computing, vol. 39, pp. 169–182 (2016)

Rapidly Convergent Integrals and Function Evaluation

Heba al Kafri, David J. Jeffrey(✉), and Robert M. Corless

Department of Applied Mathematics, The University of Western Ontario, London, ON N6G 2B3, Canada
djeffrey@uwo.ca

Abstract. We analyse integrals representing the Lambert W function, paying attention to computations using various rules. Rates of convergence are investigated, with the way in which they vary over the domain of the function being a focus. The first integral evaluates with errors independent of the function variable over a significant range. The second integral converges faster, but the rate varies with the function variable.

1 Introduction

Expressing functions as definite integrals is well established in mathematics: two obvious examples being the Gamma function and the Bessel functions. Integral representations can be used, for example, to prove properties, or membership of a function class, or for numerical evaluation. For numerical evaluation, rapid convergence is desirable. Here we examine two integrals and their suitability for numerical evaluation. We are interested in applying the rapidly converging trapezoidal rule [3,6]. We first re-examine an integral representation of Lambert W that was recently considered in [2]. We demonstrate that the convergence rate is largely independent of the function argument, except for the immediate neighborhood of the branch point. Then we consider an integral posed by Poisson [5]. The convergence rate is higher, but the rate depends upon the function argument.

2 First Integral

It was shown in [4] that $W(x)/x$ is a Stieltjes function, because it can be written as

$$f(x) = \frac{W(x)}{x} = \frac{1}{\pi}\int_{1/e}^{\infty} \frac{\phi(t)}{x+t}\,\mathrm{d}t.$$

Here, $\phi(t) = \Im W(-t)/t$. For computation, the integral is unsatisfactory because it defines W in terms of W, and numerically it is slow to converge. The self-reference can be removed by making the substitution $v = \Im W(-t)$. Then from [1, Eq. (4.1)], we have that $t = v\csc v e^{-v\cot v}$,

$$\frac{v}{t}\frac{\mathrm{d}t}{\mathrm{d}v} = 1 + v^2\csc^2 v - 2v\cot v = v^2 + (1 - v\cot v)^2, \tag{1}$$

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 270–274, 2017.
https://doi.org/10.1007/978-3-319-72453-9_20

and the integral becomes

$$K(x,v) = \frac{v^2 + (1 - v\cot v)^2}{x + v\csc v\, \exp(-v\cot v)}\ ,$$
$$f(x) = \frac{W(x)}{x} = \frac{1}{\pi}\int_0^\pi K(x,v)\,\mathrm{d}v. \tag{2}$$

Since the integrand is symmetric, $K(x,-v) = K(x,v)$, the integral can be written so that the integrand is periodic:

$$\frac{W(x)}{x} = \frac{1}{2\pi}\int_{-\pi}^{\pi} K(x,v)\,\mathrm{d}v. \tag{3}$$

In this form, it is clearly a candidate for the rapid convergence of the trapezoidal rule, observed for periodic functions and described in [6].

2.1 Trapezoidal Rule

The trapezoidal rule was applied to (3) in [2]. Translating their algorithm into Maple syntax, we have[1]

```
IaconoBoyd := proc (x, N) local t, h, j, K, S;
   h := 2*Pi/N;
   S := 0;
   K := (x, t)->((1-t*cot(t))^2+t^2)/(x+t*csc(t)*exp(-t*cot(t)));
   for j to N do
      t := -Pi+(j-1+sqrt(31/71))*h;
      S := S+evalf(K(x, t));
   end do;
   S/N ;
end proc;
```

The basic theory of trapezoidal integration is phrased in terms of periodic functions [6], and hence it may seem that all numerical schemes must be based on the interval over which the function is periodic. The integrand under consideration possesses a symmetry of which we can take advantage, and the question arises whether this will change the convergence rate. It does not. Specifically, there is no reason to avoid (2). To apply the trapezoidal rule to (2), we first note that although both $K(x,0)$ and $K(x,\pi)$ are undefined, the limits exist:

$$\lim_{v\to 0+} K(x,v) = \begin{cases} 0\ , & x \neq -1/e\ , \\ 2e\ , & x = -1/e\ , \end{cases}$$
$$\lim_{v\to \pi-} K(x,v) = 0.$$

Therefore these points can be omitted from the sum for $x \neq -1/e$. It becomes

[1] Maple programmers will be jumping to improve this code for style and efficiency, but the issue is rate of convergence, not programming efficiency, and the code reflects the algorithm as given in [2].

```
F_approx := proc (x, N) local h;
              h := evalf(Pi/N);
              add(K(x, k*h), k = 1 .. N-1)/N;
            end proc;
```

This procedure does not experience the errors reported in [2].

2.2 Convergence Rate

The function $f(x)$ is monotonically decreasing, with $f(-1/e) = e$ and $f(0) = 1$, and $f(x) \to 0$ as $x \to \infty$. We have measured the absolute error in the evaluation of this function. This differs from [2], who measured the absolute error in $xf(x) = W(x)$ which grows monotonically from -1 to ∞.

The error was computed for values of $x = -0.35 + 0.4k$ for $k = 0, 50$ and $n = 10 + 10\ell$ for $\ell = 0..40$. Computations were made close to the branch point $x = -1/e$, using $x = -0.36, -0.367, -0.3678$. Maple was used with the setting `Digits:=30`. Since the values being calculated are of $O(1)$, this setting of `Digits` corresponds to a absolute errors of 10^{-30}, and in plots of log of the error the limit of error reduction is seen to be $\ln(10^{-30}) \approx -70$. Figure 1 shows the log of the absolute error against $\sqrt{n}$ for the various x. The three points close to the branch point are plotted in green crosses, while the remainder are plotted in circles of various colours.

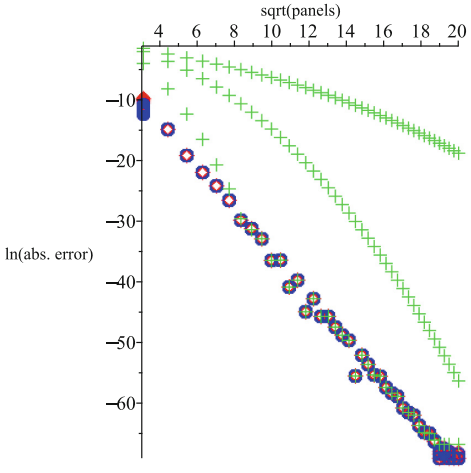


Fig. 1. A plot of $\ln|f(x) - F_{\text{approx}(x,n)}|$ against $\sqrt{n}$ for various values of x. The crosses correspond to the three values closest to the branch point. The symbols for other points are superimposed on each other. (Color figure online)

We observe that the convergence rate is practically independent of x for all points except the three closest to the branch point. Even for $x = -0.36$, the

error follows the other points for $n > 100$. In other words, the errors decay to leading order as $\alpha e^{-\beta\sqrt{n}}$ for a $\beta > 0$ that is independent of x. The data shows the error is approximately proportional to $e^{-3.5\sqrt{n}}$.

3 Second Integral

In 1823, Poisson derived an integral for W:

$$W(x) = \frac{2}{\pi}\int_0^{\pi} \frac{\cos\frac{3}{2}\theta - xe^{-\cos\theta}\cos\left(\frac{5}{2}\theta + \sin\theta\right)}{1 - 2xe^{-\cos\theta}\cos(\theta + \sin\theta) + x^2e^{-2\cos\theta}} \cos\tfrac{1}{2}\theta \, \mathrm{d}\theta, \qquad (4)$$

The integrand is periodic with period 2π, but is symmetric, and therefore the trapezoidal rule can be restricted to the interval $[0, \pi]$. The integrand is non-zero at $\theta = 0$ but can be evaluated everywhere without special considerations. Unlike the first integral, this integral is valid only in the interval $(-1/e, e)$. We plot the errors in Figs. 2 and 3. Since the integral is exactly 0 for $\theta = 0$, the error is minimum there. Consequently, we have plotted separately the cases of $x < 0$ and $x > 0$. In this case, the plots are the log of the absolute error in the estimate of $W(x)$ against n. Thus the convergence is faster than for the first integral. The behaviour with x is different from the first integral. Now to leading order the error is $\alpha e^{-\beta(x)n}$. Another contrast with the first integral is that the error maintains its functional form as the branch point is approached, albeit with a decreasing β.

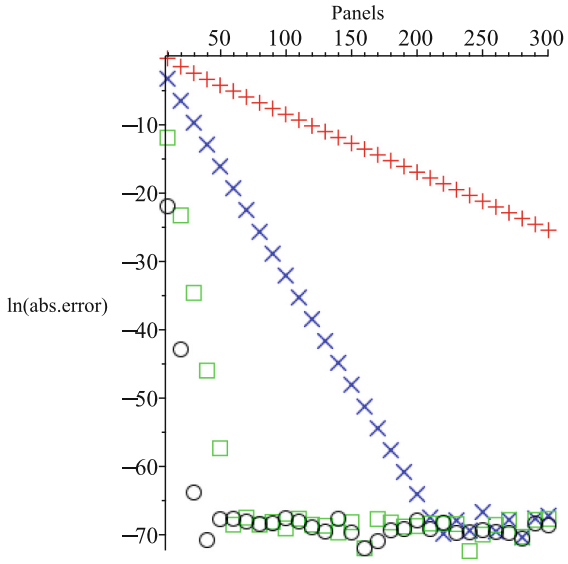


Fig. 2. A plot of the log of the absolute error against n for various positive x; the legend is + = 2.5; × = 2.0; □ = 1.0; ◯ = 0.5. The negative slope increases as x decreases to zero.

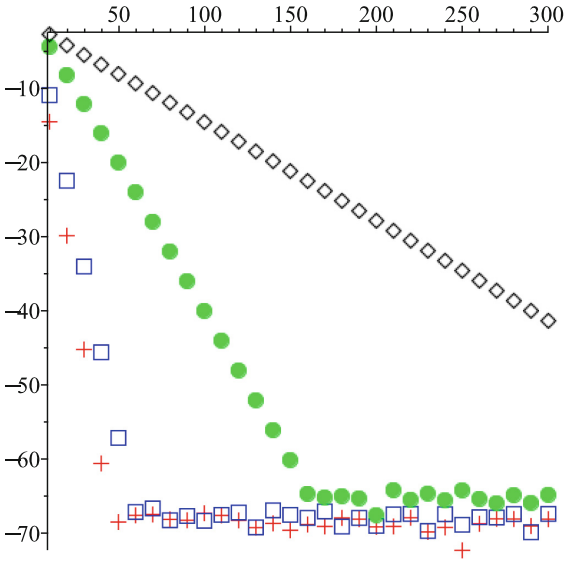


Fig. 3. A plot of the log of the absolute error against n for various negative x: $+ = -0.25$; $\square = -0.3$; $\bullet = -0.36$; $\diamond = -0.367$. The negative slope decreases as x decreases to the branch point $x = -1/e$.

References

1. Corless, R.M., Gonnet, G.H., Hare, D.E.G., Jeffrey, D.J., Knuth, D.E.: On the Lambert W function. Adv. Comput. Math. **5**, 329–359 (1996)
2. Iacono R., Boyd, J.P.: New approximations to the principal real-valued branch of the Lambert W-function. Adv. Comput. Math., 1–34 (2017)
3. Trefethen, L.N., Weideman, J.A.C.: The exponentially convergent trapezoidal rule. SIAM Rev. **56**(3), 385–458 (2014)
4. Kalugin, G.A., Jeffrey, D.J., Corless, R.M., Borwein, P.B.: Stieltjes and other integral representations for functions of Lambert W. Integr. Transforms Spec. Functions **23**(8), 581–593 (2012). https://doi.org/10.1080/10652469.2011.613830
5. Poisson, S.-D.: Suite du mémoire sur les intégrales définies et sur la sommation des séries. J. de l'École Royale Polytechnique **12**, 404–509 (1823)
6. Weideman, J.A.C.: Numerical integration of periodic functions: a few examples. Am. Math. Mon. **109**, 21–36 (2002)

Stirling Numbers, Lambert W and the Gamma Function

David J. Jeffrey(✉) and Nick Murdoch

Department of Applied Mathematics, The University of Western Ontario,
London, ON N6G 2B3, Canada
djeffrey@uwo.ca

Abstract. Stirling's asymptotic expansion for the Gamma function can be derived from an expansion of the Lambert W function about one of its branch points. Although the series expansions around this branch point have been known for some time, the coefficients in the series were only known as solutions of nonlinear recurrence relations. Here we show that the coefficients can be expressed using associated Stirling numbers.

1 Introduction

The Lambert W function is the multivalued inverse of the map $W \mapsto We^W$. The branches, denoted by W_k ($k \in \mathbb{Z}$), are defined through the equations [4]

$$\forall z \in \mathbb{C}, \qquad W_k(z)\exp(W_k(z)) = z \ , \tag{1}$$

$$W_k(z) \sim \ln_k z \text{ as } |z| \to \infty \ , \tag{2}$$

where $\ln_k z = \ln z + 2\pi i k$, and $\ln z$ is the principal branch of natural logarithm [5]. The function possesses a branch point at $z = -1/e$, where we have

$$p = \pm\sqrt{2(ez+1)} \ , \tag{3}$$

$$W_k(z) = \sum_{\ell=0}^{\infty} \mu_\ell p^\ell = -1 + p - \tfrac{1}{3}p^2 + \tfrac{11}{72}p^3 \ldots \ , \tag{4}$$

and the choice of sign determines the particular branch of W. The coefficients are given by [2,4] $\mu_0 = -1$, $\mu_1 = 1$, $\alpha_0 = 2$, and $\alpha_1 = -1$, and

$$\mu_k = \frac{k-1}{k+1}\left(\frac{\mu_{k-2}}{2} + \frac{\alpha_{k-2}}{4}\right) - \frac{\alpha_k}{2} - \frac{\mu_{k-1}}{k+1} \ , \tag{5}$$

$$\alpha_k = \sum_{j=2}^{k-1} \mu_j \mu_{k+1-j} \ . \tag{6}$$

The connection with Stirling's approximation for the Gamma, or factorial, function comes through the related expansions [1,3,7]

$$W_0\left(-\exp(-1-\tfrac{1}{2}z^2)\right) = \sum_{n\geq 0} a_n z^n \ , \tag{7}$$

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 275–279, 2017.
https://doi.org/10.1007/978-3-319-72453-9_21

where the a_n are given by $a_0 = -1$, $a_1 = 1$ and

$$a_n = \frac{-1}{(n+1)a_1}\left(a_{n-1} + \sum_{k=2}^{n-1} k a_k a_{n+1-k}\right) . \tag{8}$$

Euler's integral for n! can then be expressed using this expansion, the final result being the asymptotic series

$$n! \sim \frac{n^{n+1}}{e^n} \sum_{k\geq 0} (2k+1) a_{2k+1} \left(\frac{2}{n}\right)^{k+1/2} \Gamma\left(k+\tfrac{1}{2}\right), \tag{9}$$

where Γ is the gamma function.

We obtain expressions for a_k and μ_k, following the methods of [6]. In [6], we obtained expressions for derivatives of W around a general point (excluding the branch point), while here we choose the branch point. For an analytic function $y = f(x)$ with $f(0) = 0$, $f'(0) \neq 0$, the inverse function $x = \breve{f}(y)$ can be expressed as

$$\breve{f}(y) = \sum_{n\geq 1} a_n y^n ,$$

where

$$a_n = \frac{1}{n!} \lim_{x\to 0} \frac{d^{n-1}}{dx^{n-1}} \frac{x^n}{f(x)^n} .$$

We, however, shall use an alternative, equivalent, expression:

$$a_n = \frac{1}{n} \left[x^{n-1}\right] \left(\frac{x}{f(x)}\right)^n , \tag{10}$$

where we have used the notation of Knuth that $[x^m]g(x)$ equals the coefficient of x^m in a Taylor series expansion of $g(x)$ in the variable x.

2 First Expansion

Definition 1. *The r-associated Stirling numbers of the first kind, more briefly Stirling r-cycle numbers, are defined by the generating function*

$$\left(\ln\frac{1}{1-z} - \sum_{j=1}^{r-1} \frac{z^j}{j}\right)^m = m! \sum_{n\geq rm} \begin{bmatrix} n \\ m \end{bmatrix}_{\geq r} \frac{z^n}{n!} . \tag{11}$$

These numbers have combinatorial significance: they count the number of ways in which m cycles can be formed from n distinct objects, with each cycle required to have a minimum of r objects. The case $r = 1$ was studied by Stirling.

Theorem 1. *The coefficients a_n defined in (7)–(8) are*

$$a_n = \frac{1}{n}\sum_{k=0}^{n-1}\binom{-n/2}{k}\frac{k!\,2^k}{(n+2k-1)!}\begin{bmatrix} n+2k-1 \\ k \end{bmatrix}_{\geq 3}, \tag{12}$$

$$= \frac{1}{n!!}\sum_{k=0}^{n-1}\frac{(-1)^k}{(n+2k-1)!!}\begin{bmatrix} n+2k-1 \\ k \end{bmatrix}_{\geq 3}. \tag{13}$$

Proof. We set $W(-\exp(-1-z^2/2)) = -1 + A$, then from (1) we have

$$(-1+A)\exp(-1+A) = -\exp(-1-z^2/2)\ .$$

Simplifying and taking the logarithm of both sides, we have

$$\ln(1-A) + A = -z^2/2\ .$$

Now solving for z gives us

$$z = \sqrt{2\ln\frac{1}{1-A} - 2A}\ .$$

Lagrange inversion, using (10), gives $A = \sum_{n=1} a_n z^n$, where

$$a_n = \frac{1}{n}\left[A^{n-1}\right]\left(\frac{A}{\sqrt{2\ln\frac{1}{1-A}-2A}}\right)^n = \frac{1}{n}\left[A^{n-1}\right]\left(\frac{\ln(1/(1-A))-A}{A^2/2}\right)^{-n/2},$$

and the notation of (10) has been used. In order to connect with Stirling numbers, we rearrange the expression:

$$a_n = \frac{1}{n}\left[A^{n-1}\right]\left(1+\frac{\ln(1/(1-A))-A-A^2/2}{A^2/2}\right)^{-n/2}. \tag{14}$$

Now using the binomial expansion, we get

$$a_n = \frac{1}{n}\left[A^{n-1}\right]\sum_{k\geq 0}\binom{-n/2}{k}2^k\,k!\sum_{m=3k}\begin{bmatrix} m \\ k \end{bmatrix}_{\geq 3}\frac{v^{m-2k}}{m!}\ .$$

With further simplification, we obtain (12), which can be rewritten using standard identities to complete the theorem.

We can also express the coefficients using Stirling 2-cycle numbers.

Theorem 2. *The coefficients a_n defined in (7)–(8) are*

$$a_n = \binom{\frac{3n}{2}-1}{n-1}\sum_{k=0}^{n-1}\frac{(-1)^k(n-1)!}{(n-k-1)!}\frac{2^k}{(n+2k)!}\begin{bmatrix} n+2k-1 \\ k \end{bmatrix}_{\geq 2}, \tag{15}$$

$$= \left(\frac{n}{2}+1\right)^{\overline{n-1}}\sum_{k=0}^{n-1}\frac{(-1)^k}{(n-k-1)!}\frac{2^k}{(n+2k)!}\begin{bmatrix} n+2k-1 \\ k \end{bmatrix}_{\geq 2}. \tag{16}$$

Proof. We start from (14). We apply the binomial theorem twice.

$$a_n = \frac{1}{n}\left[A^{n-1}\right] \sum_{k\ge 0} \binom{-n/2}{k} 2^k \sum_{\ell>0} \binom{k}{\ell} (-1)^{k-\ell} \left(\frac{\ln(1/(1-A)) - A}{A^2/2}\right)^{\ell} .$$

Expanding the final term as Stirling numbers completes the proof.

3 Second Expansion

We now turn to the expansion given in (4).

Definition 2. *We define r-associated Stirling numbers of the second kind, more briefly called Stirling r-partition numbers, by*

$$\left(e^z - \sum_{j=0}^{r-1} \frac{z^j}{j!}\right)^m = m! \sum_{n\ge rm} \left\{ {n \atop m} \right\}_{\ge r} \frac{z^n}{n!} . \tag{17}$$

As with the cycle numbers defined above, these numbers have combinatorial significance. They count the number of ways in which n distinct objects can be partitioned into m subsets, which each subset having at least r members.

Theorem 3. *The coefficients μ_k defined in (4)–(6) are given by*

$$B(n,0) = 1 \tag{18}$$

$$B(n,m) = \sum_{k=1}^{m} \binom{-n/2}{k} \frac{k!\,2^k}{(m+2k)!} \left\{ {m+2k \atop k} \right\}_{\ge 3} , \qquad m>0, \tag{19}$$

$$\mu_n = \sum_{m=0}^{n-1} \frac{n^{n-m-2}}{(n-m-1)!\,2^{n-m-1}} B(n,m). \tag{20}$$

Proof. Set $W = -1 - v$. If v is assumed positive, then this describes the W_{-1} branch, which means the series in p will be all positive terms. From (3), we have $z = -e^{-1}(1 - p^2/2)$. Then $We^W = z$ becomes

$$p^2 = 2 - 2e^{-v} - 2ve^{-v}$$

So Lagrange inversion gives $v = \sum v_n p^n$ and

$$\begin{aligned}
v_n &= \frac{1}{n}[v^{n-1}] \left(\frac{\sqrt{2-2e^{-v}-2ve^{-v}}}{v}\right)^{-n} \\
&= \frac{1}{n}[v^{n-1}] e^{nv/2} \left(1 + \frac{e^v - 1 - v - v^2/2}{v^2/2}\right)^{-n/2} \\
&= \frac{1}{n}[v^{n-1}] e^{nv/2} \sum_{k=0} \binom{-n/2}{k} \frac{2^k}{v^{2k}} k! \sum_{m=3k} \left\{ {m \atop k} \right\}_{\ge 3} \frac{v^m}{m!} \\
&= \frac{1}{n}[v^{n-1}] e^{nv/2} \sum_{k=0} \binom{-n/2}{k} 2^k k! \sum_{m=k} \left\{ {m+2k \atop k} \right\}_{\ge 3} \frac{v^m}{(m+2k)!}
\end{aligned}$$

There is still the factor $e^{nv/2}$ to be included. Expanding this and collecting terms completes the theorem.

4 Concluding Remarks

We noted in the introduction that the expansion coefficients are related to Stirling's approximation to $n!$ [7]. It is convenient to substitute form (13) into (9) and use the result

$$\Gamma\left(k+\frac{1}{2}\right) = \frac{(2k-1)!!}{2^k}\sqrt{\pi}\ ,$$

to obtain

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \sum_{k\geq 0}\frac{1}{(2n)^k}\sum_{\ell=0}^{2k}\frac{(-1)^\ell}{2^\ell\,(\ell+k)!}\begin{bmatrix}2k+2\ell\\ \ell\end{bmatrix}_{\geq 3}\ . \tag{21}$$

A similar expression can be obtained using 2−cycle numbers from the expressions in Theorem 2.

The present results, together with those in [6], replace recurrence relations with explicit expressions in terms of known combinatorial numbers. The expressions in the paper have all been programmed in Maple, where they can be written in a single line, in contrast to the recurrence relations, which take several lines ad run more slowly. The "Stirling's approximation" considered here is in fact one of several variations existing in the literature [1], and it would be interesting to see whether the coefficients in other forms can also be found in terms of Stirling numbers. Substituting the expressions found here into the previously known recurrence relations yields combinatorial identities which are challenging to prove independently.

References

1. Borwein, J.M., Corless, R.M.: The Gamma function in the Monthly, American Math Monthly, in press. arXiv:1703.05349 [math.HO]
2. Coppersmith, D.: Personal communication
3. Copson, E.T.: An Introduction to the Theory of Functions of a Complex Variable. The Clarendon Press, Oxford (1935)
4. Corless, R.M., Gonnet, G.H., Hare, D.E.G., Jeffrey, D.J., Knuth, D.E.: On the lambert W function. Adv. Comput. Math. **5**, 329–359 (1996)
5. Jeffrey, D.J., Hare, D.E.G., Corless, R.M.: Unwinding the branches of the Lambert W function. Math. Sci. **21**, 1–7 (1996)
6. Jeffrey, D.J., Kalugin, G.A., Murdoch, N.: Lagrange inversion and Lambert W. In: SYNASC 2015 Proceedings, pp 42–46. IEEE Computer Society (2015)
7. Marsaglia, G., Marsaglia, J.C.: A new derivation of Stirling's approximation to $n!$. Am. Math. Monthly **97**, 826–829 (1990)

The Potential and Challenges of CAD with Equational Constraints for SC-Square

James H. Davenport[1] and Matthew England[2(✉)]

[1] Department of Computer Science, University of Bath, Bath, UK
J.H.Davenport@bath.ac.uk
[2] Faculty of Engineering, Environment and Computing,
Coventry University, Coventry, UK
Matthew.England@coventry.ac.uk

Abstract. Cylindrical algebraic decomposition (CAD) is a core algorithm within Symbolic Computation, particularly for quantifier elimination over the reals and polynomial systems solving more generally. It is now finding increased application as a decision procedure for Satisfiability Modulo Theories (SMT) solvers when working with non-linear real arithmetic. We discuss the potentials from increased focus on the logical structure of the input brought by the SMT applications and SC^2 project, particularly the presence of equational constraints. We also highlight the challenges for exploiting these: primitivity restrictions, well-orientedness questions, and the prospect of incrementality.

1 Introduction

1.1 Cylindrical Algebraic Decomposition

The original aim of Cylindrical Algebraic Decomposition (CAD), as introduced by [Col75], was *Quantifier Elimination* (QE). More precisely, given

$$Q_{k+1}x_{k+1}\dots Q_n x_n \Phi(x_1,\dots,x_n) \tag{1}$$

where $Q_i \in \{\forall,\exists\}$ and Φ is a Tarski Formula; produce an equivalent formula, $\Psi(x_1,\dots,x_k)$ which is quantifier free. Here a *Tarski Formula* is a Boolean combination of predicates $f_j\,\sigma_j, 0$ with $\sigma_j \in \{=,\neq,>,\geq,<,\leq\}, f_j \in \mathbf{Q}[x_1,\dots,x_n]$.

The CAD of [Col75] was a major breakthrough, with a running time "merely" doubly-exponential in n, as opposed to previous methods [Tar51].

The methodology of Collins' CAD is broadly as follows:

1. Retaining from (1) only the f_j (call this set S_n) and the order of the x_i, compute a CAD of $\mathbf{R}^n$, sign-invariant for the f_i.
   (a) Repeatedly project $S_\ell \subset \mathbf{Q}[x_1,\dots,x_\ell]$ to $S_{\ell-1} := P_C(S_\ell) \subset \mathbf{Q}[x_1,\dots,x_{\ell-1}]$ where P_C is Collins' projection operator.
   (b) Isolate real roots of S_1 to produce a CAD of $\mathbf{R}^1$ sign-invariant for S_1.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 280–285, 2017.
https://doi.org/10.1007/978-3-319-72453-9_22

(c) Repeatedly lift the decomposition of $\mathbf{R}^{\ell-1}$ to one of $\mathbf{R}^{\ell}$, sign-invariant for S_ℓ. To lift over a cell in $\mathbf{R}^{\ell-1}$ we substitute a sample point of the cell into S_ℓ; perform univariate root isolation and decompose accordingly. P_C is chosen so that the sample point is representative of the whole cell.

2. Using the Q_i and Φ identify cells of the induced CAD of $\mathbf{R}^k$ true for (1).
3. Deduce Ψ.

There have been many improvements since [Col75]: we quote only two here, referring to [BDE+16] for a more detailed summary.

[McC84]**:** This replaced the operator P_C by a much smaller operator P_M, simultaneously replacing "sign-invariant" by "order-invariant" in Step 1. However, the system has to be "well-oriented", which can only be seen with hindsight, when a lack of it manifests itself by a polynomial being nullified, i.e. vanishing entirely, over a cell of dimension > 0.

[Laz94]**:** This replaced the operator P_M by a slightly smaller operator P_L and significantly modified the lifting procedure, simultaneously replacing "order-invariant" by what is now called "Lazard-valuation-invariant" in Step 1. A gap in the proof of [Laz94] was soon spotted. It was rectified recently by [MH16], but using the technology of order-invariant and under the well-oriented restriction. A complete resolution, in terms of Lazard invariance, has been presented in the preprint [MPP17].

1.2 New Applications: SC²

The authors are involved in the EU Project SC² which aims to forge interaction between the communities of **S**ymbolic **C**omputation and **S**atisfiability **C**hecking [SC2]. CAD and QE are traditionally found in the former but recently the technology behind them have been applied in Satisfiability Modulo Theory (SMT) solvers [JdM12, for example] where the problem is usually not to perform full QE but to test satisfiability, finding either a witness point or (minimal) proof of unsatisfiability. Such solvers are used routinely in industries such as software verification. The problem sets are different to those typical in CAD: often lower degree polynomials but far more of them and in more variables. Viewed from Satisfiability Checking the CAD procedure outlined above is curious, particular in its discarding of the logical structure in Step 1.

2 Potentials

2.1 Equational Constraint

The fact that the σ_j and Φ are essentially ignored in Step 1 was noticed in [Col98], at least for the special case

$$\Phi(x_1, \ldots, x_n) \equiv F_1(x_1, \ldots, x_n) = 0 \wedge \Phi'(x_1, \ldots, x_n) \tag{2}$$

(where F_1 depends non-trivially on x_n and is primitive): intuitively the key idea is that we do not care about the polynomials in Φ' away from $F_1 = 0$.

We refer to $F_1 = 0$ as an *equational constraint* (more generally, an equation implied by the formula). This was formalised in [McC99]. The key result there is the following.

Theorem 1 ([McC99, Theorem 2.2])**.** *Let $r > 2$, let $f(x_1, \ldots, x_r)$ and $g(_1, \ldots, x_r)$ be real polynomials of positive degrees in the main variable x_r, let $R(x_1, \ldots, x_{r-1})$ be the resultant of f and g, and suppose that $R \neq 0$. Let S be a connected subset of $\mathbf{R}^{r-1}$ on which f is delineable and in which R is order-invariant. Then g is sign-invariant in each section of f over S.*

In the context of (2) this justifies replacing $P_M(S_r)$ by the reduced projection operator

$$P_M(F_1; S_r) := P_M(\{F_1\}) \cup \{\mathrm{Res}_{x_r}(F_1, f_i) : f_i \in \Phi'\}, \tag{3}$$

at least for the first projection. If S_r has n polynomials of degree d, $P_M(S_r)$ has $\frac{1}{2}n(n+1)$ polynomials of degree $O(d^2)$ whereas $P_M(F; S_r)$ has n such.

2.2 Multiple Equational Constrains

If there are multiple equational constraints then it is possible to use a variant (slightly enlarged) of the reduced operator (3) for projections beyond the first. The idea is to *propagate* the constraints by noticing their resultant is also implied by the formula but does not contain the main variable [McC01].

More recently, in [EBD15] the present authors identified savings in the lifting phase: the fact that Theorem 1 provides not just delineability but sign-invariance for g means there is no need to isolate and decompose with respect to the real roots of g. This, combined with the use of Gröbner Basis technology to control the degree growth of projection polynomials allowed us to present an improved complexity analysis of CAD with multiple equational constraints in [ED16]. Broadly speaking, we decrease the double exponent by one for each equational constraint.

2.3 Equational Constraints of Sub-formulae

If instead of (2), our problem has the form

$$\Phi(x_1, \ldots, x_n) \equiv (f_1 = 0 \wedge \Phi_1) \vee (f_2 = 0 \wedge \Phi_2) \vee \cdots, \tag{4}$$

then we can write it in the form (2) by letting $F_1 = \prod f_i$. However, as was observed in [BDE+13], we can do better by analysing the inter-dependencies in (4) more carefully, building a truth-table invariant CAD (TTICAD) for the collection of sub-formulae. Intuitively the key idea is that we do not care about the polynomials in Φ_i outside $f_i = 0$. TTICAD was expanded in [BDE+16] to the case where not every disjunct has an equation (so there is no overall equational constraint for Φ).

3 Challenges

Section 2 identifies a wealth of technology for making greater use of the logical structure of the CAD input. However, there are a number of challenges.

3.1 Need for Primitivity

All the theory of reduced projection operators requires that the constraint be primitive. No technology currently exists (beyond reverting to sign-invariance) for the non-primitive case (although ideas were sketched in [EBD15]). Note that the restriction is not just on the input but also constraints found through propagation. In [DE16] the Davenport-Heinz examples [DH88] used to demonstrate the doubly exponential complexity of CAD were shown to lack primitivity, showing that the non-primitive case is genuinely more difficult.

3.2 Well-Orientedness

All the existing theory of reduced projection operators rests on the mathematics of order-invariance developed for P_M. The reduced operators not only require this condition of P_M but actually extend it (they are less complete). The lack of this condition is only discovered at the end of CAD (when we lift with respect to the offending polynomials). For traditional CAD this means a large waste of resources starting the calculation again.

As described in Sect. 1 there is a new sign-invariant projection operator P_L which achieves the savings of P_M without sacrificing completeness. It may be possible to expand this to a family of reduced operators, but this requires development of the corresponding Lazard valuation invariance theory.

3.3 Incremental CAD

A key requirement for the effective use of CAD by SMT-solvers is that the CAD be incremental: that polynomials can be added and removed to the input with the data structures of the CAD edited rather than recalculated. Such incremental CAD algorithms are now under development by the SC^2 project [SC2].

These could offer a partial solution to the difficulties of well-orientedness. i.e. if a particular operator is found to not be well-oriented at the end of a CAD calculation the next step would be to revert to a less efficient operator which is usually a superset of the original one. Hence we could edit the existing decomposition to take into account these additional polynomials.

However, the use of CAD with equational constraints incrementally may exhibit strange behaviour in the SMT context. For example, removing a constraint that was equational could actually grow the output CAD since it necessitates the use of a larger projection operator. Correspondingly, adding an equational constraint could allow a smaller operator and shrink the output. It is not clear how SMT solvers heuristics should be adapted to handle these possibilities.

Acknowledgements. The authors are supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No H2020-FETOPEN-2015-CSA 712689 (SC^2).

References

[SC2] Ábrahám, E., et al.: SC^2: satisfiability checking meets symbolic computation. In: Kohlhase, M., Johansson, M., Miller, B., de Moura, L., Tompa, F. (eds.) CICM 2016. LNCS (LNAI), vol. 9791, pp. 28–43. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42547-4_3

[BDE+13] Bradford, R.J., Davenport, J.H., England, M., McCallum, S., Wilson, D.J.: Cylindrical algebraic decompositions for boolean combinations. In: Proceedings of ISSAC 2013, pp. 125–132 (2013). https://doi.org/10.1145/2465506.2465516

[BDE+16] Bradford, R.J., Davenport, J.H., England, M., McCallum, S., Wilson, D.J.: Truth table invariant cylindrical algebraic decomposition. J. Symb. Comput. **76**, 1–35 (2016). https://doi.org/10.1016/j.jsc.2015.11.002

[Col75] Collins, G.E.: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Proceedings of 2nd GI Conference Automata Theory & Formal Languages, pp. 134–183 (1975). https://doi.org/10.1007/3-540-07407-4_17

[Col98] Collins, G.E.: Quantifier elimination by cylindrical algebraic decomposition — twenty years of progess. In: Caviness, B.F., Johnson, J.R. (eds.) Quantifier Elimination and Cylindrical Algebraic Decomposition. TEXTSMONOGR, pp. 8–23. Springer, Wien (1998). https://doi.org/10.1007/978-3-7091-9459-1_2

[DE16] Davenport, J.H., England, M.: Need polynomial systems be doubly-exponential? In: Greuel, G.-M., Koch, T., Paule, P., Sommese, A. (eds.) ICMS 2016. LNCS, vol. 9725, pp. 157–164. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42432-3_20

[DH88] Davenport, J.H., Heintz, J.: Real quantifier elimination is doubly exponential. J. Symb. Comput. **5**(1–2), 29–35 (1988). https://doi.org/10.1016/S0747-7171(88)80004-X

[EBD15] England, M., Bradford, R., Davenport, J.H.: Improving the use of equational constraints in cylindrical algebraic decomposition. In: Proceedings of ISSAC 2015, pp. 165–172. ACM (2015). https://doi.org/10.1145/2755996.2756678

[ED16] England, M., Davenport, J.H.: The complexity of cylindrical algebraic decomposition with respect to polynomial degree. In: Gerdt, V.P., Koepf, W., Seiler, W.M., Vorozhtsov, E.V. (eds.) CASC 2016. LNCS, vol. 9890, pp. 172–192. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45641-6_12

[JdM12] Jovanović, D., de Moura, L.: Solving non-linear arithmetic. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS (LNAI), vol. 7364, pp. 339–354. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31365-3_27

[Laz94] Lazard, D.: An improved projection operator for cylindrical algebraic decomposition. In: Proceedings of Algebraic Geometry and its Applications (1994). https://doi.org/10.1007/978-1-4612-2628-4_29

[McC84] McCallum, S.: An Improved Projection Operation for Cylindrical Algebraic Decomposition. Ph.D. thesis, University of Wisconsin-Madison Computer Science (1984)

[McC99] McCallum, S.: On projection in CAD-based quantifier elimination with equational constraints. In: Proceedings of ISSAC 1999, pp. 145–149 (1999). https://doi.org/10.1145/309831.309892

[McC01] McCallum, S.: On propagation of equational constraints in CAD-based quantifier elimination. In: Proceedings of ISSAC 2001, pp. 223–231. ACM (2001). https://doi.org/10.1145/384101.384132

[MH16] McCallum, S., Hong, H.: On using Lazard's projection in CAD construction. J. Symb. Comput. **72**, 65–81 (2016). https://doi.org/10.1016/j.jsc.2015.02.001

[MPP17] McCallum, S., Parusinski, A., Paunescu, L.: Arxiv (2017). https://arxiv.org/abs/1607.00264v2

[Tar51] Tarski, A.: A decision method for elementary algebra and geometry. University of California Press (1951). In: Caviness, B.F., Johnson, J.R. (eds.) Quantifier Elimination and Cylindrical Algebraic Decomposition. TEXTSMONOGR, pp. 24–84. Springer, Vienna (1998) (Republished). https://doi.org/10.1007/978-3-7091-9459-1_3

Combinatorics and Codes in Computer Science

New Small 4-Designs with Nonabelian Automorphism Groups

Vedran Krčadinac[1] and Mario Osvin Pavčević[2(✉)]

[1] Department of Mathematics, Faculty of Science, University of Zagreb, Bijenička 30, 10000 Zagreb, Croatia
[2] Department of Applied Mathematics, Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia
mario.pavcevic@fer.hr

Abstract. We prove the existence of designs with parameters 4-(18, 9, 56), 4-(18, 9, 70), 4-(19, 9, 84), and 4-(19, 9, 105) by assuming suitable automorphism groups and performing a computer search using the method of Kramer and Mesner.

Keywords: Combinatorial design · Automorphism group
Kramer-Mesner method

1 Introduction

It is a well-known fact that explicit construction of discrete structures can be a hard task, because of the combinatorial explosion of possibilities that need to be checked in an exhaustive search. If the structures under investigation fulfil many properties, they tend to be rare in the search space and this often makes the search infeasible. One approach is to add further constraints and thus reduce the size of the search space, all in hope that our sought structures satisfy these constraints.

In this paper we perform a search for combinatorial designs with rather strong conditions. A t-(v, k, λ) *design* $\mathcal{D}$ consists of a v-element set of *points* $\mathcal{P}$ together with a collection $\mathcal{B}$ of k-element subsets called *blocks*, such that every t-element subset of points is contained in exactly λ blocks. If we denote the number of blocks by $b := |\mathcal{B}|$, it follows immediately that $b = \lambda\binom{v}{t}/\binom{k}{t}$. The design is said to be *simple* if $\mathcal{B}$ contains no repeated blocks. A table of parameter sets for which the existence of simple t-(v, k, λ) designs has been established can be found in [4]. New existence results appear in [1,2,6,8–10]. Our challenge was to fill some of the remaining gaps and find some missing designs by computer search. We were successful for parameters 4-(18, 9, 56), 4-(18, 9, 70), 4-(19, 9, 84), and 4-(19, 9, 105).

This work has been fully supported by the Croatian Science Foundation under the project 1637.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 289–294, 2017.
https://doi.org/10.1007/978-3-319-72453-9_23

An *automorphism* of $\mathcal{D}$ is a permutation of points leaving $\mathcal{B}$ invariant. The set of all automorphisms forms a group under composition, the *full automorphism group* $\mathrm{Aut}(\mathcal{D})$. The additional constraint in our construction procedure will be a prescribed subgroup $G \leq \mathrm{Aut}(\mathcal{D})$. We shall assume automorphism groups which allow us either to perform an exhaustive search, or to find at least one design and thus settle the existence problem. A short description of the construction method and some computational details are given in the next section. Our results are presented in Sects. 3 and 4.

2 Construction Method

Our natural search space consists of families of size b containing k-element subsets of the point set $\mathcal{P} = \{1, \ldots, v\}$. Therefore, its size is $\binom{\binom{v}{k}}{b}$. Let G be given as a permutation group on the set $\mathcal{P}$. This action induces the action of G on the subsets of $\mathcal{P}$ as well. Let $\mathcal{T}_1, \ldots, \mathcal{T}_r$ and $\mathcal{K}_1, \ldots, \mathcal{K}_c$ be the orbits of G on t-element subsets and on k-element subsets of $\mathcal{P}$, respectively. Our task is to choose some block orbits, the union of which forms $\mathcal{B}$ and satisfies the λ-balancing property. Note that the group action might shrink the search space down to size $\binom{\binom{v}{k}/|G|}{b/|G|}$.

Let a_{ij} be the number of subsets $K \in \mathcal{K}_j$ containing a given $T \in \mathcal{T}_i$. This number does not depend on the choice of T. The $r \times c$ matrix $A = [a_{ij}]$ is the *Kramer-Mesner matrix*. Clearly, the a_{ij}'s corresponding to the chosen block orbits need to sum up row-wise to λ. In other words, simple t-(v, k, λ) designs with G as an automorphism group exist if and only if the system of linear equations $A \cdot x = \lambda j$ has 0–1 solutions $x \in \{0, 1\}^c$, where $j = (1, \ldots, 1)^\tau$ is the all-one vector of length r. This is the celebrated method of construction introduced by Kramer and Mesner in [5]. It has been used thereafter to find many new designs with prescribed automorphism groups.

We use GAP [3] to compute the orbits $\mathcal{T}_i$, $\mathcal{K}_j$ and the matrix A. The second step of the construction, computationally the most challenging one, is to find solutions of the Kramer-Mesner system $A \cdot x = \lambda j$. Solving systems of linear equations over $\{0, 1\}$ is a known NP complete problem. We use Wasserman's solver [11] based on the LLL algorithm. Finally, to decide whether the constructed designs are isomorphic and to compute their full automorphism groups, we use *nauty/Traces* by McKay and Piperno [7].

The choice of the automorphism group G is crucial for the construction to be successful. Designs with parameters t-(v, k, λ) often have a cyclic automorphism of order $u \in \{v-2, v-1, v\}$, but in the cases of our interest this assumption does not reduce the search space sufficiently. We shall therefore assume a nonabelian semidirect product $G \cong \mathbb{Z}_u.\mathbb{Z}_w$ as automorphism group.

3 Designs with 18 Points

In this section we consider the existence of 4-$(18, 9, \lambda)$ designs. A necessary condition is that λ is divisible by $\lambda_{\min} = 14$, i.e. of the form $\lambda = 14\,\mathrm{m}$. Simple

designs can exist only for $\lambda \leq \lambda_{\max} = \binom{v-t}{k-t} = \binom{14}{5}$. The *supplementary design* with $\mathcal{B}' = \binom{\mathcal{P}}{k} \setminus \mathcal{B}$ as the set of blocks is a simple 4-$(18, 9, \lambda_{\max} - \lambda)$ design, and therefore is suffices to consider designs with $\lambda = 14\,\mathrm{m} \leq \lambda_{\max}/2$. We shall denote the largest m satisfying this inequality by M; in this case we have $M = 71$.

According to the Handbook of Combinatorial Designs [4], simple 4-$(18, 9, 14\,\mathrm{m})$ designs exist for $m \in \{3, 6, \ldots, 71\}$. We shall prove their existence for $m \in \{4, 5\}$ by prescribing automorphisms $\alpha = (1, 2, \ldots, 17)$ and

$$\beta = (2, 4, 10, 11, 14, 6, 16, 12, 17, 15, 9, 8, 5, 13, 3, 7).$$

They generate a nonabelian group $G_1 = \langle \alpha, \beta \rangle$ isomorphic to the semidirect product $\mathbb{Z}_{17}.\mathbb{Z}_{16}$ of order 272.

Theorem 1. *Up to isomorphism there are exactly* 38 *simple* 4-$(18, 9, 70)$ *designs with* G_1 *as automorphism group.*

Proof. The group G_1 generates $r = 14$ orbits on 4-element subsets and $c = 190$ orbits on 9-element subsets of $\mathcal{P} = \{1, \ldots, 18\}$. The Kramer-Mesner system $A \cdot x = \lambda j$ is of size 14×190 and can be solved exhaustively by the solver [11] in less than a minute of CPU time. There are 38 solutions; using nauty [7] we checked that the corresponding designs are non-isomorphic and that G_1 is their full automorphism group.

Generally, it may happen that different solutions of the Kramer-Mesner system correspond to isomorphic designs. The number of isomorphism classes is usually smaller than the number of solutions. The fact that the opposite occurs in Theorem 1 is due to G_1 being the full automorphism group of the 38 designs and the normalizer of G_1 in the symmetric group S_{18} being G_1 itself. Any isomorphism mapping one design onto another is contained in the normalizer of the full automorphism group of the second design. Because of $N_{S_{18}}(G_1) = G_1$, the 38 designs can only be mapped onto themselves.

We present one of the 4-$(18, 9, 70)$ designs by listing its base blocks in Table 1. The design is the union of the corresponding G_1-orbits. Group generators and base blocks for all 38 designs from Theorem 1, as well as for other designs constructed in this paper are available on the web page

https://web.math.pmf.unizg.hr/~krcko/results/newdesigns.html

Table 1. Base blocks for a 4-$(18, 9, 70)$ design with G_1.

$\{1, 2, 3, 4, 5, 6, 7, 9, 11\}$, $\{1, 2, 3, 4, 5, 6, 8, 11, 16\}$, $\{1, 2, 3, 4, 5, 6, 9, 13, 18\}$,
$\{1, 2, 3, 4, 6, 7, 8, 9, 12\}$, $\{1, 2, 3, 4, 5, 7, 11, 12, 16\}$, $\{1, 2, 3, 4, 5, 7, 8, 14, 18\}$,
$\{1, 2, 3, 4, 5, 8, 10, 14, 18\}$, $\{1, 2, 3, 5, 11, 13, 14, 15, 18\}$

We turn now to the $m = 4$ case, i.e. to simple 4-$(18, 9, 56)$ designs. The Kramer-Mesner system from the proof of Theorem 1 has no solutions for $\lambda = 56$,

and thus these designs with G_1 as automorphism group do not exist. We look for examples with a smaller automorphism group $G_2 = \langle \alpha, \beta^2 \rangle \cong \mathbb{Z}_{17}.\mathbb{Z}_8$ of order 136. Now there are $r = 28$ orbits of 4-element subsets and $c = 380$ orbits of 9-element subsets. The corresponding 28×380 Kramer-Mesner system is too large to be solved exhaustively, but after several weeks of CPU time the solver [11] found one solution. This proves the following theorem.

Theorem 2. *Simple* 4-$(18, 9, 56)$ *designs with* G_1 *as automorphism group do not exist. There is at least one such design with* G_2 *as automorphism group.*

Base blocks for this design are given in Table 2.

Table 2. Base blocks for a 4-$(18, 9, 56)$ design with G_2.

$\{1, 2, 4, 8, 9, 12, 13, 15, 17\}$,	$\{1, 2, 6, 8, 9, 10, 13, 15, 17\}$,	$\{1, 2, 8, 9, 10, 12, 15, 17, 18\}$,
$\{1, 5, 8, 9, 11, 12, 15, 17, 18\}$,	$\{1, 4, 5, 6, 8, 9, 10, 15, 17\}$,	$\{1, 4, 5, 8, 9, 11, 15, 17, 18\}$,
$\{1, 5, 6, 8, 9, 11, 12, 15, 16\}$,	$\{1, 5, 6, 8, 9, 14, 15, 17, 18\}$,	$\{1, 5, 8, 9, 10, 15, 16, 17, 18\}$,
$\{1, 3, 4, 8, 9, 10, 12, 15, 17\}$,	$\{1, 6, 7, 8, 9, 11, 15, 17, 18\}$,	$\{1, 8, 9, 11, 12, 14, 15, 17, 18\}$

The existence of 4-$(18, 9, 14\,\mathrm{m})$ designs remains open for $m \in \{1, 2\}$. We checked that these designs do not allow G_1 or G_2 as automorphism groups, nor several other permutation groups roughly of the same size.

4 Designs with 19 Points

Using the same notation as in the previous section, for 4-$(19, 9, \lambda)$ designs we have $\lambda = 21\,\mathrm{m}$, $\lambda_{\min} = 21$, $\lambda_{\max} = \binom{15}{5}$, and $M = 71$. According to the Handbook [4], these designs exist for $m \in \{3, 6, \ldots, 71\}$. Once more we can settle the gap $m \in \{4, 5\}$. Let $\gamma = (1, 2, \ldots, 19)$ and

$$\delta = (2, 3, 5, 9, 17, 14, 8, 15, 10, 19, 18, 16, 12, 4, 7, 13, 6, 11)$$

be generators of the group $G_3 = \langle \gamma, \delta \rangle$ of order 342 isomorphic to the semidirect product $\mathbb{Z}_{19}.\mathbb{Z}_{18}$. The corresponding Kramer-Mesner system is of size 14×280. It seems within reach for an exhaustive search, but at the time of this writing the solver [11] is still running. So far it found more than 100000 solutions.

Theorem 3. *Up to isomorphism there are at least* 100000 *simple* 4-$(19, 9, 105)$ *designs with* G_3 *as automorphism group.*

All of the found designs have G_3 as their full automorphism group and $N_{S_{19}}(G_3) = G_3$ holds, and thus different solutions correspond to non-isomorphic designs. Interestingly, simple 4-$(19, 9, 105)$ designs can also be constructed from the group G_1 from the previous section. In that case the Kramer-Mesner system is of size 18×360 and we performed a partial search. The solver [11] found some solutions fairly quickly.

Theorem 4. *There is at least one simple* 4-$(19, 9, 105)$ *design with* G_1 *as automorphism group.*

Base blocks for one design for each of the groups G_1 and G_3 is given in Table 3.

Table 3. Base blocks for 4-$(19, 9, 105)$ designs with G_1 and G_3.

$\{1,2,3,4,5,6,8,9,13\}$, $\{1,2,3,4,5,6,8,9,18\}$, $\{1,2,3,4,5,6,8,13,14\}$, $\{1,2,3,4,5,6,9,12,14\}$, $\{1,2,3,4,5,6,10,12,18\}$, $\{1,2,3,4,5,7,8,10,18\}$, $\{1,2,3,4,5,7,11,12,16\}$, $\{1,2,3,4,5,7,14,18,19\}$, $\{1,2,3,4,5,7,16,18,19\}$, $\{1,2,3,4,5,8,9,13,19\}$, $\{1,2,3,4,5,8,15,18,19\}$, $\{1,2,3,4,5,10,13,18,19\}$, $\{1,2,3,4,6,7,9,10,19\}$, $\{1,2,3,4,6,9,13,15,19\}$, $\{1,2,3,5,11,13,14,15,18\}$, $\{1,2,3,5,11,13,14,15,19\}$
$\{1,2,3,4,5,6,7,10,15\}$, $\{1,2,3,4,5,6,7,12,15\}$, $\{1,2,3,4,5,6,9,11,12\}$, $\{1,2,3,4,5,6,10,13,14\}$, $\{1,2,3,4,5,7,8,9,16\}$, $\{1,2,3,4,5,7,11,14,15\}$, $\{1,2,3,4,5,8,9,11,13\}$, $\{1,2,3,4,5,8,9,16,17\}$, $\{1,2,3,4,5,8,10,12,14\}$, $\{1,2,3,4,5,8,13,15,16\}$, $\{1,2,3,4,6,7,9,10,11\}$, $\{1,2,3,4,6,8,13,14,17\}$

Similarly as before, in the $m = 4$ case the group G_3 does not give rise to designs. Simple 4-$(19, 9, 84)$ designs can either be constructed non-exhaustively from the group G_1, or from a subgroup of G_3. The group $G_4 = \langle \gamma, \delta^2 \rangle \cong \mathbb{Z}_{19}.\mathbb{Z}_9$ of order 171 leads to a Kramer-Mesner system of size 24×546. The solver [11] found some solutions of both systems quickly, but the searches were not completed.

Theorem 5. *Simple* 4-$(19, 9, 84)$ *design with* G_3 *as automorphism group do not exist. There is at least one such design with* G_1 *as automorphism group and at least one such design with* G_4 *as automorphism group.*

Proof. Base blocks are given in Table 4.

Table 4. Base blocks for 4-$(19, 9, 84)$ designs with G_1 and G_4.

$\{1,2,3,4,5,6,7,8,13\}$, $\{1,2,3,4,5,6,8,10,11\}$, $\{1,2,3,4,5,6,9,10,12\}$, $\{1,2,3,4,5,6,9,15,18\}$, $\{1,2,3,4,5,7,10,12,18\}$, $\{1,2,3,4,5,7,14,15,19\}$, $\{1,2,3,4,5,7,16,18,19\}$, $\{1,2,3,4,5,8,9,10,18\}$, $\{1,2,3,4,5,8,15,18,19\}$, $\{1,2,3,4,5,10,13,18,19\}$, $\{1,2,3,4,6,9,11,15,19\}$, $\{1,2,3,4,6,10,12,16,19\}$, $\{1,2,3,4,9,11,13,18,19\}$
$\{1,2,3,4,5,8,9,12,17\}$, $\{1,2,3,4,5,9,12,13,15\}$, $\{1,2,3,5,6,7,9,10,18\}$, $\{1,2,3,5,6,8,9,14,17\}$, $\{1,2,3,5,6,12,14,17,19\}$, $\{1,2,3,5,7,8,9,11,19\}$, $\{1,2,3,5,7,8,9,14,19\}$, $\{1,2,3,5,7,9,13,14,16\}$, $\{1,2,3,5,8,9,10,13,17\}$, $\{1,2,3,5,8,9,12,13,14\}$, $\{1,2,3,5,9,10,12,14,16\}$, $\{1,2,3,5,9,10,15,17,18\}$, $\{1,2,3,5,9,11,12,15,16\}$, $\{1,2,3,5,9,11,13,14,17\}$, $\{1,2,3,5,9,11,13,15,17\}$, $\{1,2,3,5,9,11,13,15,18\}$, $\{1,2,3,5,9,11,14,15,16\}$, $\{1,2,3,6,9,10,15,17,19\}$

To conclude, the designs with $v = 19$ points constructed in this section allow automorphism groups of the form $\mathbb{Z}_u.\mathbb{Z}_w$ for $u \in \{v-2, v\}$. The designs with $v = 18$ points from the previous section allow such automorphism groups for $u = v - 1$. We also tried to construct 4-$(19, 9, 21\,\mathrm{m})$ designs for $m \in \{1, 2\}$, but failed to find suitable automorphism groups. The existence of these designs remains open.

References

1. Araya, M., Harada, M.: Mutually disjoint Steiner systems $S(5, 8, 24)$ and 5-$(24, 12, 48)$ designs. Electron. J. Comb. **17**, #N1 (2010)
2. Araya, M., Harada, M., Tonchev, V.D., Wassermann, A.: Mutually disjoint designs and new 5-designs derived from groups and codes. J. Comb. Des. **18**(4), 305–317 (2010)
3. The GAP Group, GAP - Groups, Algorithms, and Programming, Version 4.8.7 (2017). http://www.gap-system.org
4. Khosrovshahi, G.B., Laue, R.: t-designs with $t \geq 3$. In: Colbourn, C.J., Dinitz, J.F. (eds.) The Handbook of Combinatorial Designs, 2nd edn., pp. 79–101. Chapman & Hall/CRC, Boca Raton (2007)
5. Kramer, E.S., Mesner, D.M.: t-designs on hypergraphs. Discrete Math. **15**(3), 263–296 (1976)
6. Krčadinac, V.: Some new designs with prescribed automorphism groups. J. Comb. Des. (2017, to appear)
7. McKay, B.D., Piperno, A.: Practical graph isomorphism II. J. Symbolic Comput. **60**, 94–112 (2014)
8. Östergård, P.R.J., Pottonen, O.: There exists no Steiner system $S(4, 5, 17)$. J. Comb. Theory Ser. A **115**(8), 1570–1573 (2008)
9. Stinson, D.R., Swanson, C.M., van Trung, T.: A new look at an old construction: constructing (simple) 3-designs from resolvable 2-designs. Discrete Math. **325**, 23–31 (2014)
10. van Trung, T.: Simple t-designs: a recursive construction for arbitrary t. Des. Codes Cryptogr. **83**(3), 493–502 (2017)
11. Wassermann, A.: Finding simple t-designs with enumeration techniques. J. Comb. Des. **6**(2), 79–90 (1998)

On Classifying Steiner Triple Systems by Their 3-Rank

Dieter Jungnickel[1], Spyros S. Magliveras[2(✉)], Vladimir D. Tonchev[3], and Alfred Wassermann[4]

[1] Mathematical Institute, University of Augsburg, 86135 Augsburg, Germany
jungnickel@math.uni-augsburg.de
[2] Department of Mathematical Sciences,
Florida Atlantic University, Boca Raton, FL, USA
spyros@fau.edu
[3] Department of Mathematical Sciences,
Michigan Technological University, Houghton, MI 49931, USA
tonchev@mtu.edu
[4] Mathematical Institute, University of Bayreuth, 95440 Bayreuth, Germany
alfred.wassermann@uni-bayreuth.de

Abstract. It was proved recently by Jungnickel and Tonchev (2017) that for every integer $v = 3^{m-1}w$, $m \geq 2$, and $w \equiv 1, 3 \pmod 6$, there is a ternary linear $[v, v-m]$ code C, such that every Steiner triple system STS(v) on v points and having 3-rank $v-m$, is isomorphic to an STS(v) supported by codewords of weight 3 in C. In this paper, we consider the ternary $[3^n, 3^n - n]$ code C_n ($n \geq 3$), that supports representatives of all isomorphism classes of STS(3^n) of 3-rank $3^n - n$. We prove some structural properties of the triple system supported by the codewords of C_n of weight 3. Using these properties, we compute the exact number of distinct STS(27) of 3-rank 24 supported by the code C_3. As an application, we prove a lower bound on the number of nonisomorphic STS(27) of 3-rank 24, and classify up to isomorphism all STS(27) supported by C_3 that admit a certain automorphism group of order 3.

1 Introduction

A *block design* (or simply a design) D is a pair of a finite set X of v *points* and a collection $\mathcal{B}$ of b subsets of X called *blocks* [3,6]. In this paper we consider only designs that have no repeated blocks. Given non-negative integers $t \leq k \leq v$, λ, a t-(v, k, λ) design $D = (X, \mathcal{B})$ is a design with v points and blocks of size k such that every t-subset of X is contained in exactly λ blocks. The *incidence matrix* $A = (a_{ij})$ of a design D is a $b \times v$ $(0,1)$-matrix with columns labeled by the points and rows labeled by the blocks, where $a_{i,j} = 1$ if the ith block contains the jth point, and $a_{i,j} = 0$ otherwise. Two designs are *isomorphic* if there is a bijection between their point sets that induces a bijection of the block sets. An *automorphism* of a design is any isomorphism of the design to itself. The set of all automorphisms of D forms the automorphism group $\mathrm{Aut}(D)$ of D.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 295–305, 2017.
https://doi.org/10.1007/978-3-319-72453-9_24

A linear $[n, k]$ code C over a finite field $\mathrm{GF}(q)$ is a k-dimensional vector subspace of $\mathrm{GF}(q)^n$ [4]. A generator matrix of C is a $k \times n$ matrix whose rows form a basis of C, and a parity check matrix of C is an $(n-k) \times n$ matrix whose rows form a basis of the null space (or dual code) $C^\perp$ of C. An automorphism of a code C is any permutation on the n coordinates which preserves C. The Hamming weight of a vector is the number of its nonzero components. The support $supp(x)$ of a vector $x = (x_1, \ldots, x_n)$ is the set of the coordinate indices of its nonzero components.

A typical way to associate a block design D with a linear $[n, k]$ code C is by considering the n code coordinates, $\{1, 2, \ldots, n\}$, as point set, and the supports of codewords of a given Hamming weight w as blocks [2,22]. In such a case, we refer to D as a design supported by C.

A triple system is a block design with blocks of size 3, and a Steiner triple system is a 2-$(v, 3, 1)$ design. We will use the notation $\mathrm{STS}(v)$ for a Steiner triple system on v points. An $\mathrm{STS}(v)$ exists if and only if $v \equiv 1, 3 \pmod 6$, and the number of nonisomorphic $\mathrm{STS}(v)$ grows exponentially with linear growth of v [6, II.2.4]. The largest value of v for which all $\mathrm{STS}(v)$ have been classified up to isomorphism is $v = 19$ [11].

If p is a prime number, the p-rank of a design D is defined as the rank of its incidence matrix over the finite field of order p, $\mathrm{GF}(p)$.

In [8], Doyen, Hubaut and Vandensavel proved that the p-rank of an $\mathrm{STS}(v)$ with $v \geq 7$ can be smaller than v only if $p = 2$ or $p = 3$, and showed how the 2-ranks and 3-ranks of Steiner triple systems can be determined in terms of projective and affine hyperplanes, which were studied by Teirlinck [17,18]. In particular, it was proved in [8] that the 3-rank of any Steiner triple system D on 3^n points is greater than or equal to $3^n - n - 1$, and equality holds if and only if D is isomorphic to the classical Steiner triple system $\mathrm{AG}_1(n, 3)$ of the points and lines in the n-dimensional affine geometry $\mathrm{AG}(n, 3)$.

In [1], Assmus proved that for every $v \equiv 1, 3 \pmod 6$, $v = 2^k \cdot u - 1 > 7$, u odd, and any integer i in the range $1 \leq i < k$, there is a binary $[v, v - k + i]$ code C such that every $\mathrm{STS}(v)$ having 2-rank $v - k + i$ is supported by C. As an application of this result, explicit formulas for the exact total number of distinct $\mathrm{STS}(2^n - 1)$ of 2-rank $2^n - n$, as well as the exact total number of distinct 3-$(2^n, 4, 1)$ designs of 2-rank $2^n - n$, were found in [20,21]. The formula from [20] was used in [15] to classify up to isomorphism all STS(31) of 2-rank 27.

A ternary analogue of Assmus's result was proved recently in [9], showing that for every $v = 3^{m-1}w$, $m \geq 2$, and $w \equiv 1, 3 \pmod 6$, there is a ternary $[v, v - m]$ code C, such that every $\mathrm{STS}(v)$ of 3-rank $v - m$ is isomorphic to an $\mathrm{STS}(v)$ supported by C. In particular, the ternary $[3^n, 3^n - n - 1]$ code spanned by the incidence matrix of the classical $\mathrm{STS}(3^n)$ of the points and lines in $\mathrm{AG}(n, 3)$, has a parity check matrix H_n of the form (1),

$$H_n = \begin{pmatrix} 1 \ldots 1 \\ B \end{pmatrix}, \tag{1}$$

where B is an $n \times 3^n$ matrix whose column set consists of all distinct vectors in $\mathrm{GF}(3)^n$. Furthermore, if $1 \leq i \leq n$, and B_i is an $(n - i) \times 3^n$ matrix obtained by deleting i rows of B, then the $(n + 1 - i) \times 3^n$ matrix

$$H_{n,i} = \begin{pmatrix} 1 \dots 1 \\ B_i \end{pmatrix} \tag{2}$$

is the parity check matrix of a ternary $[3^n, 3^n - n - 1 + i]$ code that contains representatives of all isomorphism classes of STS(3^n) having 3-rank smaller than or equal to $3^n - n - 1 + i$ [9].

We note that the column set of the $(n - i) \times 3^n$ matrix B_i in (2) consists of all vectors of $\mathrm{GF}(3)^{n-i}$, where each vector appears exactly 3^i times as a column of B_i [9]. Thus, the matrix B_i, and consequently, $H_{n,i}$, is unique up to a permutation of columns.

In this paper, we consider the ternary $[3^n, 3^n - n]$ code C_n with a parity check matrix $H = H_{n,1}$ that supports representatives of all isomorphism classes of STS(3^n) of 3-rank $3^n - n$. Since the only (up to isomorphism) STS(3^2) is the affine plane of order 3, having 3-rank equal to $6 = 3^2 - 2 - 1$, we will assume that $n \geq 3$. We prove some structural properties of the triple system supported by the codewords of C_n of weight 3. Using these properties, we compute the exact number of distinct STS(27) of 3-rank $3^3 - 3 = 24$ supported by the code C_3. As an application, we prove a lower bound on the number of nonisomorphic STS(27) of 3-rank 24, and classify up to isomorphism all STS(27) supported by C_3 which admit a certain automorphism of order 3.

In [10], general bounds for the number of STS($2^n - 1$) and STS(3^n) with prescribed rank are given, which confirm our results.

2 The Triple System Supported by C_n

For convenience of notation, we will assume that the columns of the parity check matrix $H = H_{n,1}$ of C_n (cf. (2)), are ordered lexicographically. The columns of the submatrix B_1 of H can be viewed as points of the $(n-1)$-dimensional affine geometry $\mathrm{AG}(n-1, 3)$, each point repeated three times. Thus, the columns of B_1 with indices $3j+1, 3j+2, 3j+3$, $j = 0, 1, \dots, 3^{n-1}$ are three identical vectors being the ternary presentation of j, and H has the following form:

$$H = \begin{pmatrix} 111\ 111 \dots 111 \\ 000\ 000 \dots 222 \\ \dots\ \dots\ \dots\ \dots \\ 000\ 111 \dots 222 \end{pmatrix}. \tag{3}$$

Theorem 1. *(i) The code C_n is invariant under a group G which is a wreath product of two groups G_1 and G_2, where G_1 is the direct product of 3^{n-1} copies of the symmetric group S_3, where each copy of S_3 acts on a triple of identical columns of the parity check matrix H, and G_2 is the collineation group of the $(n-1)$-dimensional affine geometry* $\mathrm{AG}(n-1, 3)$.

(ii) The order of G is equal to

$$6^{3^{n-1}} \cdot 3^{n-1}(3^{n-1} - 1)(3^{n-1} - 3) \cdots (3^{n-1} - 3^{n-2}). \tag{4}$$

Proof. The statements of the theorem follow directly from the structure of the column set of the parity check matrix H as given in (3). We note that the columns of the $(n-1) \times 3^n$ submatrix of H obtained by deleting the first all-one row, has as columns the vectors representing the points of $\mathrm{AG}(n-1,3)$, each point repeated three times.

A block design $D = (X, B)$ is a *group divisible design* with parameters λ_1, λ_2 if the point set X is a union of pairwise disjoint subsets G_i, called *groups*, such that every two distinct points that belong to the same group appear together in λ_1 blocks, while every two points that belong to different groups appear together in λ_2 blocks.

Theorem 2. *(i) The number of codewords of weight* 3 *in* C_n *is*

$$3^{n-1}(3^{n+1} - 7). \tag{5}$$

(ii) The triple system D_n *supported by the codewords of* C_n *of weight* 3 *is a group divisible design with groups* G_i $(1 \le i \le 3^{n-1})$ *of size* 3*, and parameters* $\lambda_1 = 1$, $\lambda_2 = 3$.

(iii) The automorphism group of D_n *coincides with the automorphism group of* C_n.

(iv) D_n *is an* $1\text{-}(3^n, 3, (3^{n+1} - 7)/2)$ *design.*

(v) If $B' = \{p_1, p_2, p_3\}$, $B'' = \{p_1, p_2, p_4\}$ *are two distinct blocks sharing two points* p_1, p_2*, then* p_1, p_2 *belong to two different groups* G_{i_1}, G_{i_2}, $i_1 \neq i_2$, *and* p_3 *and* p_4 *belong to a third group* G_{i_3}, $i_3 \neq i_1$, $i_3 \neq i_2$.

Proof. (i) Consider the following partition of the columns of $H = (h_1, h_2, \ldots, h_{3^n})$ (cf. (3)) in 3^{n-1} groups G_i of size 3:

$$G_1 = \{h_1, h_2, h_3\}, G_2 = \{h_4, h_5, h_6\}, \ldots, G_{3^{n-1}} = \{h_{3^n-2}, h_{3^n-1}, h_{3^n}\}. \tag{6}$$

Every group G_i $(1 \le i \le 3^{n-1})$ consists of three identical columns, and columns belonging to different groups are different and label different points of $\mathrm{AG}(n-1,3)$. Let $x = (x_1, \ldots, x_n)$ be a codeword of weight 3 with nonzero components x_i, x_j, x_k. Since x is orthogonal to the all-one vector, being the first row of the parity check matrix H, we have $x_i = x_j = x_k$. Without loss of generality, we may assume $x_i = x_j = x_k = 1$. We claim that either (a) the indices i, j, k label three columns of H belonging to the same group, G_s, or (b) label columns belonging to three different groups, $G_{s_i}, G_{s_j}, G_{s_k}$.

To see this, we first note that the three columns in any given group obviously determine a codeword of weight 3. Now suppose that the two groups G_{s_i} and G_{s_j} are the same, while the third index k determines a different group G_{s_k}. Subtracting x from the codeword determined by the group G_{s_i} then gives a codeword y of weight 2, with one of its two nonzero entries being a 1 in the third column belonging to G_{s_i}, and the other nonzero entry a 2 in a column belonging to the different group G_{s_k}. Then y has to be orthogonal to all rows of H, which

clearly implies that the vector of length 3^{n-1} and weight 2 with entries 1 in position s_i and 2 in position s_k is orthogonal to the matrix H' obtained by using just one of the three columns in each group, that is, the matrix H_{n-1} in (1). As H_{n-1} is the parity check matrix for the code determined by the lines in the affine space $\mathrm{AG}(n-1,3)$, this contradicts the well-known fact that the minimum weight vectors in this code have weight 3 (they are the vectors associated with the lines of this space); see, for instance, [19, Sect. 3]. Thus (b) is the only alternative to (a), as claimed.

In case (a), there are 3^{n-1} codewords with $x_i = x_j = x_k = 1$, and 3^{n-1} codewords with $x_i = x_j = x_k = 2$. Thus, the total number of codewords of weight 3 obtained in case (a) is

$$2 \cdot 3^{n-1}. \tag{7}$$

The three columns of H labeled by the support of a codeword in the second case (b) correspond to a line in the affine space $\mathrm{AG}(n-1,3)$. The number of lines in $\mathrm{AG}(n-1,3)$ is $3^{n-2}(3^{n-1}-1)/2$. Since every point of $\mathrm{AG}(n-1,3)$ is labeled by three identical columns of H, the number of codewords of weight 3 obtained in case (b) is

$$3^3 \cdot 3^{n-2}(3^{n-1}-1). \tag{8}$$

Adding (7) and (8) and simplifying completes the proof of part (i).

In what follows, we will refer to a codeword of weight 3 as being of type (a) or (b) respectively.

(ii) The groups of D_n correspond to the groups G_i $(1 \le i \le 3^{n-1})$ (cf. (6)) of triples of identical columns of H. As seen from the proof of part (i), every pair of points $\{p_i, p_j\}$, where p_i, p_j are the indices of two columns of H belonging to a group G_i, appears in exactly one block $B = \{p_i, p_j, p_k\}$, where p_k is the index of the third column in G_i. Thus, $\lambda_1 = 1$, and the groups (6) determine a set S of 3^{n-1} pairwise disjoint blocks (or a *parallel class*) that must belong to every $\mathrm{STS}(3^n)$ supported by the code C_n. Equivalently, the blocks in S are supports of codewords of weight 3 of type (a) from the proof of part (i).
Let now p_i, p_j be two points of D_n belonging to two different blocks B', B'' from S. If $L = \{p_i, p_j, p_k\}$ is a block containing p_i and p_j, then p_k belongs to a block B''' from S, different from B' and B''. It follows that L is the support of a codeword of type (b), and there are exactly two more blocks that contain p_i, and p_j, namely $\{p_i, p_j, p_m\}$ and $\{p_i, p_j, p_u\}$, where $B''' = \{p_k, p_m, p_u\}$. Thus, $\lambda_2 = 3$, and the proof of (ii) is complete.

(iii) It was proved in [9] that for every $n \ge 3$, the code C_n supports representatives of all nonisomorphic $\mathrm{STS}(3^n)$ having 3-rank equal to $3^n - n$. In addition, it was proved in [9,17], that for every $n \ge 3$ and every integer r in the range $3^n - n - 1 \le r \le 3^n - 1$, there exists an $\mathrm{STS}(3^n)$ having 3-rank equal to r. Consequently, for every $n \ge 3$, the code C_n contains an $\mathrm{STS}(3^n)$ of 3-rank $3^n - n$. This implies that the code C_n is spanned by the set of all codewords of weight 3. Since every codeword of weight 3 has all

its nonzero entries equal to either 1 or 2, it follows that C_n coincides with the linear span over GF(3) of the rows of the incidence matrix of D_n, and the statement (iii) follows.

(iv) By Theorem 2, part (iii), and Theorem 1, the automorphism group of D_n acts transitively on the set of points, hence D_n is a 1-design. Thus, the number of blocks containing a given point is equal to $3b/3^n$, where b is the number of blocks of D_n. By part (i) $b = (3^{n-1}(3^{n+1} - 7))/2$, and the statement (iv) follows.

(v) Any two blocks that share two points must be the supports of codewords of type (b), and the statement (v) follows from the proofs of (i) and (ii).

3 Counting STS(27) Supported by the Code C_3

The known lower bound 10^{11} on the number of nonisomorphic STS(27) [13, II.1.3] indicates that a complete classification up to isomorphism of all such designs is computationally infeasible. Various classes of STS(27) possessing some additional properties or symmetry have been investigated and classified, for example, all STS(27) possessing a transitive automorphism group have been classified in [7].

In this section, we use properties of the triple system D_n proved in Theorem 2 to compute the exact number of distinct STS(27) supported by the code C_3.

By Theorem 2, part (i), the code C_3 contains exactly 666 codewords of weight 3. The set of minimum weight codewords consists of 333 $(0, 1)$-codewords, that can be viewed as the rows of an incidence matrix of the triple system D_3, and the remaining 333 codewords are $(0, 2)$-vectors, being scalar multiples of the rows of the incidence matrix of D_3. An incidence matrix of D_3 is available at

http://www.math.mtu.edu/~tonchev/dn333.txt

The incidence matrix of any STS(27) supported by C_3 consists of 117 $(0, 1)$-codewords of weight 3, sharing pairwise at most one nonzero positions, or equivalently, a constant weight code C' with 117 codewords of weight 3 and Hamming distance 4. We note that the code C' is an optimal constant weight code meeting the unrestricted Johnson bound.

We define a graph Γ_{333} having as vertices the blocks of D_3, where two blocks are adjacent if they share at most one point. Then every STS(27) supported by C_3 is a maximal clique of size 117 in STS(27). An attempt to find and store all 117-cliques using Cliquer [14] crashes quickly due to the enormous number of 117-cliques in Γ_{333} which fill up all available RAM memory of every computer we were able to use.

Another approach to find all STS(27) is by using a $\binom{27}{2} \times 333$ Kramer-Mesner matrix B with rows labeled by the 2-subsets of the point set $\{1, 2, \ldots, 27\}$, and columns labeled by the blocks of D_3, with an entry equal to 1 if the corresponding 2-subset is contained in the corresponding block, and 0 otherwise. Any STS(27) in C_3 corresponds to a set of 117 columns of B whose sum is the all-one column of size 351. We used a program implementing this approach written by the

second author in APL and a parallel version of the "dancing links" algorithm by Knuth [12]. It took approximately one day on the computer cluster of the University of Bayreuth to count all solutions without storing.

A much more elegant approach that allowed us to count all STS(27) in the code C_3 easily on a single computer is based on a simple, but as it turned out, very helpful observation.

Recall that by Theorem 2, (ii), every pair of points belonging to some of the nine blocks $B_1 = \{1, 2, 3\}, B_2 = \{4, 5, 6\}, \ldots, B_9 = \{25, 26, 27\}$ from the parallel class S, defined by the groups G_i, $1 \le i \le 9$, appear exactly in one block of D_3, while any two points which belong to different blocks B_i, B_j, $1 \le i < j \le 9$, appear together in three blocks. By Theorem 2, (iv), D_3 is a 1-design with every point appearing in 37 blocks.

We consider the partition of the set of 333 blocks of D_3 into two parts, $D_{1,2,3}$, and $\bar{D}_{1,2,3}$, where $D_{1,2,3}$ is the set of 109 blocks of D_3 containing at least one of the points 1, 2, 3, and $\bar{D}_{1,2,3}$ is the set of the remaining 224 blocks of D_3, that is, the blocks that do not contain any of the points 1, 2, 3.

Let $\Gamma_{1,2,3}$ and $\bar{\Gamma}_{1,2,3}$ be the induced subgraphs of Γ_{333} on $D_{1,2,3}$ and $\bar{D}_{1,2,3}$ respectively. If $B = \{x, y, z\}$ is a block of an STS(27), there are exactly 37 blocks that contain at least one of the points x, y, z: the block B itself, plus three disjoint sets of 12 blocks each, passing through one of the points x, y, and z, respectively.

Thus, every STS(27) supported by C_3 has 37 blocks from $D_{1,2,3}$, and the remaining 80 blocks are from $\bar{D}_{1,2,3}$. By Theorem 2, (v), every block of D_3 from $D_{1,2,3}$ meets every block of D_3 from $\bar{D}_{1,2,3}$ in at most one point. Consequently, every vertex of $\Gamma_{1,2,3}$ is adjacent in Γ_{333} to every vertex of $\bar{\Gamma}_{1,2,3}$. Thus, we have the following.

Lemma 1. *(1) Every 117-clique in Γ_{333} is a union of a 37-clique in $\Gamma_{1,2,3}$ with an 80-clique in $\bar{\Gamma}_{1,2,3}$.*

(2) The total number of 117-cliques in Γ_{333} is equal to the product of the number of 37-cliques in $\Gamma_{1,2,3}$ and the number of 80-cliques in $\bar{\Gamma}_{1,2,3}$.

Luckily, Cliquer [14] was able to compute quickly on a personal computer with 8 GB RAM memory, that $\Gamma_{1,2,3}$ contains exactly 20,736 37-cliques, and $\bar{\Gamma}_{1,2,3}$ contains exactly 429,981,696 80-cliques.

Thus, we have the following.

Theorem 3. *The total number of distinct* STS(27) *supported by the code C_3 is equal to*

$$N_3 = 8,916,100,448,256. \tag{9}$$

We consider two designs D', D'' as distinct, if at least one block of D' is not a block of D''.

The code C_3 contains STS(27) having 3-rank 24 or 23. Any STS(27) of 3-rank 23 is isomorphic to the design $\mathrm{AG}_1(3, 3)$ of lines in AG(3, 3) [8].

We can identify an STS(27) of 3-rank 23 supported by C_3 as follows. We extend the parity check matrix H to a matrix H' obtained by adding one extra row to H, having entries equal to 0, 1, 2 in every three positions corresponding

to a group G_i $(1 \leq i \leq 9)$. The matrix H' is a parity check matrix of a ternary $[27, 23]$ code C' being the span of the incidence matrix of an STS(27) isomorphic to $A_1(3, 3)$ (this is a special case of a more general result from [9,19]). We note that H' is a generator matrix of a ternary $[27, 4]$ code equivalent to the generalized first order Reed-Muller code of length 27 over GF(3).

The code C' has minimum weight 3, and the supports of the minimum weight codewords form an STS(27) isomorphic to $\mathrm{AG}_1(3, 3)$. Each codeword of weight 3 in C' is a codeword of weight 3 in C_3, which is orthogonal to the extra row of H'.

After identifying 117 rows of the 333×27 incidence matrix A of D_3 that form the incidence matrix of a design D' being an STS(27) of 3-rank 23, we add one extra column to A, having 1's in the rows corresponding to the blocks of D', and 0's elsewhere, and then compute with Magma [5] the automorphism group of the resulting 333×28 matrix. This group G', being the intersection of the automorphism group of D' with the automorphism group of D_3, is of order 23,328.

Dividing the group order of the automorphism group of D_3 (cf. Theorem 1) by the order of G' gives the exact number of STS(27) of 3-rank 23 that are supported by C_3.

Theorem 4. *The number of* STS(27) *of* 3*-rank* 23 *supported by* C_3 *is*

$$186,624. \tag{10}$$

Theorems 3 and 4 imply the following.

Theorem 5. *The number of* STS(27) *of* 3*-rank* 24 *supported by* C_3 *is*

$$N_{3,24} = 8,916,100,261,632. \tag{11}$$

According to the results of [9], the code C_3 contains representatives of all nonisomorphic STS(27) of 3-rank 24. Classifying all STS(27) of 3-rank 24 up to isomorphism will be the subject of future work.

Let m denote the number of the nonisomorphic STS(27) of 3-rank 24 supported by C_3, and let $E_1, \ldots, E_m$ be a set of m pairwise nonisomorphic STS(27) of 3-rank 24. Then we have the obvious equation

$$N_{3,24} = \sum_{i=1}^{m} \frac{|G|}{|\operatorname{Aut}(E_i)|},$$

where $|G| = 4,353,564,672$ is the order of the automorphism group of D_3 and $\operatorname{Aut}(E_i)$ denotes the order of the automorphism group of E_i.

Using the trivial lower bound $|\operatorname{Aut}(E_i)| \geq 1$ implies the following lower bound on the number of nonisomorphic STS(27) of 3-rank 24.

Theorem 6. *The number of nonisomorphic* STS(27) *of* 3*-rank* 24 *is greater than or equal to* 2048.

4 A Class of STS(27) Invariant Under a Group of Order 3

The computational complexity of classifying STS(27) supported by C_3 up to isomorphism, dramatically reduces by assuming that the designs have the additional property of being invariant under an automorphism of order 3. We illustrate this by choosing the permutation

$$f = (1, 2, 3)(4, 5, 6)...(25, 26, 27)$$

as such an automorphism, that acts regularly on each of the groups G_i of the group divisible design D_3. The 333 blocks of D_3 are partitioned into 117 orbits under the action of the cyclic group of order 3 generated by f: 108 orbits of length 3 and 9 fixed blocks, being the blocks $B_1, \ldots, B_9$, corresponding to the groups G_i, $1 \leq i \leq 9$.

We define a graph Γ_{117} having as vertices the 117 orbits under $\langle f \rangle$, where two orbits are adjacent if every two blocks belonging to the union of these orbits meet each other in at most one point. Cliquer finds quickly that the graph Γ_{117} contains exactly 531,441 45-cliques, and correspondingly, 531,441 distinct STS(27) invariant under $\langle f \rangle$ that are supported by the code C_3. To classify these 531,441 distinct STS(27) up to isomorphism, we used the normalizer $N = N_{S_{27}}(\langle f \rangle)$ of $\langle f \rangle$ in the symmetric group S_{27}, and the stabilizer of D_3 in N, whose actions split the solutions into subclasses of isomorphic designs, which were subsequently divided into 11 isomorphism classes of STS(27). For these computations, we used DESIGN Package (Version 1.6) for GAP developed by Soicher [16], as well as special algorithms developed by the authors. The results of this partial classification are formulated in the following theorem.

Table 1. Summary table of the 11 isomorphism types of STS(27)s admitting an automorphism of order 3.

Iso class	$\lvert G \rvert$	Point orbit lengths	Block orbit lengths
1	303,264	27	117
2	11,664	27	9,108
3	972	9,18	3,6,9,18,81
4	972	9,18	3,6,9,18,81
5	972	9,18	3,6,9,18,81
6	972	9,18	3,6,9,18,81
7	432	3,24	1,8,36,72
8	216	3,12,12	1,4,4,18,18,36,36
9	216	3,12,12	1,4,4,18,18,36,36
10	162	9,9,9	3,3,3,27,27,27,27
11	108	3,6,6,12	1,2,2,4,9,9,18,18,18,36

Theorem 7. *(i) The code C_3 contains eleven isomorphism classes of* STS(27) *invariant under the group of order* 3 *generated by f. (ii) One of these eleven designs has* 3*-rank* 23*, hence is isomorphic to* $\mathrm{AG}_1(3,3)$*. The remaining ten designs are of* 3*-rank* 24 *and have full automorphism groups of the following orders:* 11,664 *(one design),* 972 *(four designs),* 432 *(one design),* 216 *(two designs),* 162 *(one design), and* 108 *(one design).*

We note that the design with full group of order 11,664 is isomorphic to a design 27z9z3.14 invariant under a transitive automorphism group found in [7].

Table 1 contains information about the point and block orbit lengths of the 11 designs. Block orbit representatives of the eleven designs and data about their groups is available at

http://www.math.mtu.edu/~tonchev/sts27f3

Acknowledgements. The authors wish to thank the unknown referees for reading carefully the manuscript and making several useful remarks. Vladimir Tonchev acknowledges support by the Alexander von Humboldt Foundation and NSA Grant H98230-16-1-0011.

References

1. Assmus Jr., E.F.: On 2-ranks of Steiner triple systems. Electron. J. Comb. **2** (1995). #R9
2. Assmus Jr., E.F., Key, J.D.: Designs and Their Codes. Cambridge University Press, New York (1992)
3. Beth, T., Jungnickel, D., Lenz, H.: Design Theory, 2nd edn. Cambridge University Press, Cambridge (1999)
4. Betten, A., Braun, M., Fripertinger, H., Kerber, A., Kohnert, A., Wassermann, A.: Error-Correcting Linear Codes, Classification by Isometry and Applications. Springer, Heidelberg (2006)
5. Bosma, W., Cannon, J.: Handbook of Magma Functions. Department of Mathematics, University of Sydney, Sydney (1994)
6. Colbourn, C.J., Dinitz, J.F. (eds.): Handbook of Combinatorial Designs, 2nd edn. Chapman & Hall/CRC, Boca Raton (2007)
7. Colbourn, C.J., Magliveras, S.S., Mathon, R.A.: Transitive Steiner and Kirkman triple systems of order 27. Math. Comp. **58**(197), 441–450 (1992)
8. Doyen, J., Hubaut, X., Vandensavel, M.: Ranks of incidence matrices of Steiner triple systems. Math. Z. **163**, 251–259 (1978)
9. Jungnickel, D., Tonchev, V.D.: On Bonisoli's theorem and the block codes of Steiner triple systems. Des. Codes Crypt. (2017). https://doi.org/10.1007/s10623-017-0406-9
10. Jungnickel, D., Tonchev, V.D.: Counting Steiner triple systems with classical parameters and prescribed rank, ArXiv e-prints, 1709.06044 (2017)
11. Kaski, P., Östergård, P.R.J.: The Steiner triple systems of order 19. Math. Comp. **73**, 2075–2092 (2004)
12. Knuth, D.E.: Dancing links. In: Roscoe, A.W., Davies, J., Woodcock, J. (eds.) Millennial Perspectives in Computer Science, Cornerstones of Computing, pp. 187–214. Palgrave, Hertfordshire (2000)

13. Mathon, R., Rosa, A.: 2-(v, k, λ) designs of small order. In: Colbourn, C.J., Dinitz, J.F. (eds.) Handbook of Combinatorial Designs, 2nd edn. Chapman & Hall/CRC, Boca Raton (2007)
14. Niskanen, S., Östergård, P.R.J.: Cliquer User's Guide, Version 1.0. Technical Report T48, Communications Laboratory, Helsinki University of Technology, Espoo, Finland (2003)
15. Osuna, O.P.: There are 1239 Steiner triple systems STS(31) of 2-rank 27. Des. Codes Crypt. **40**, 187–190 (2006)
16. Soicher, L.H.: DESIGN Package (Version 1.6) for GAP. http://www.maths.qmul.ac.uk/~leonard/designtheory.org/software/
17. Teirlinck, L.: Combinatorial structures, Ph.D. thesis, University of Brussels, Brussels (1976)
18. Teirlinck, L.: On projective and affine hyperplanes. J. Comb. Theor. Ser. A **28**, 290–306 (1980)
19. Tonchev, V.D.: Linear perfect codes and a characterization of the classical designs. Des. Codes Crypt. **17**, 121–128 (1999)
20. Tonchev, V.D.: A mass formula for Steiner triple systems STS$(2^n - 1)$ of 2-rank $2^n - n$. J. Comb. Theor. Ser. A **95**, 197–208 (2001)
21. Tonchev, V.D.: A formula for the number of Steiner quadruple systems on 2^n points of 2-rank $2^n - n$. J. Comb. Des. **11**, 260–274 (2003)
22. Tonchev, V.D.: Codes and designs. In: Pless, V.S., Huffman, W.C. (eds.) Handbook of Coding Theory, Chap. 15, vol. II, pp. 1229–1267. Elsevier, Amsterdam (1998)

Right-Justified Characterization for Generating Regular Pattern Avoiding Permutations

Phan Thuan Do[1], Thi Thu Huong Tran[2(✉)], and Vincent Vajnovszki[3]

[1] Department of Computer Science, Hanoi University
of Science and Technology, 01 Dai Co Viet, Hanoi, Vietnam
thuan.dophan@hust.edu.vn
[2] Vietnamese-German University,
Le Lai street, Hoa Phu ward, Thu Dau Mot City, Binh Duong, Vietnam
huong.ttt@vgu.edu.vn
[3] LE2I UMR-CNRS 5158, Université de Bourgogne,
B.P. 47 870, 21078 Dijon Cedex, France
vvajnov@u-bourgogne.fr

Abstract. ECO-method and its corresponding succession rules allow to recursively define and construct combinatorial objects. The induced generating trees can be coded by corresponding pattern avoiding permutations. We refine succession rules by using succession functions in case when avoided patterns are regular or c-regular. Although regular patterns are hard to be recognized in general, we give a characterization for its right-justified property which is a prerequisite in the definition of the regular pattern. Based on this characterization, we show the (c-)regularity for various classes of permutations avoiding sets of patterns with variable lengths. Last, the technique of succession functions permits to construct general recursive generating models for classes of (c-) regular pattern avoiding permutations, which are constant amortized time for all classes mentioned in the paper.

Keywords: c-regular patterns · Exhaustive generating
Pattern avoiding permutations · Right-justified patterns
Regular patterns · Variable length patterns

1 Introduction

The field of pattern avoiding permutations has been showing an increasing interest in the last two decades. A very powerful way to define and describe such class of permutations is the ECO method for enumerations and recursive constructions of combinatorial object classes [1,4]. This is a recursive description of a combinatorial object class which explains how an object of size n can be reached from one and only one object of smaller size. More precisely, it presents a system of *succession rules* which induces a *generating tree* such that each node can be labeled by the number of its successors, and the set of succession rules assigns to

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 306–319, 2017.
https://doi.org/10.1007/978-3-319-72453-9_25

each node the label of its successors (see for example [2,7,9,10,14]). Occasionally, a generating tree can be encoded by one or more classes of pattern avoiding permutations in which each tree level is encoded by permutations of the same length. In this case, we can say that the succession rule of the tree corresponds to each of these classes. The root is coded by the identity permutation with length one. Let α be an n-length permutation in a generating tree; each successor is obtained from α by inserting $(n+1)$ into certain positions also known as the *active sites* of α. Notice that the sites are numbered from right to left, from 1 to $(n+1)$. We refer to some previous work for more details on specific combinatorial objects [3,5,6,11,12,18].

In the algorithmic sense, the number of nodes in a generating tree is usually proportional to the number of recursive calls of the exhaustive procedure for the corresponding permutations. However, an inserting operation into an active site of a permutation may cause a lot of switch operations inside. In order to build an efficient generating algorithm, one wishes to apply the inserting operation only at the rightmost site of a permutation, and other permutations are obtained sequentially by switching two consecutive positions in its neighbor permutation. Therefore, active sites should be aligned by consecutive sites from the rightmost one. This is defined as *right-justified* property for a pattern which is a prerequisite in the concept of *regular patterns* first mentioned in [13]. We redefine this concept by using the right-justified property and generalize to the concept of *color regular pattern* (c-regular pattern for short). One can remark that checking the regularity of a pattern set is often an exhaustive routine since we have to take into account the number of active sites of each node as well as the number of active sites of each of its successors. Although it is hard to characterize regular patterns, it is possible to characterize right-justified patterns (Proposition 2). This somehow helps to reduce the exhaustive routine by replacing the fact of finding all active sites of a permutation by determining only the maximum active site. Nevertheless, most popular right-justified patterns in the literature are either regular or c-regular. Notice that the idea of color labeling for succession rules has been mentioned in [3,6].

Last, based on the right-justified property, we use an extension definition of succession rule, call *succession function*, to derive two general exhaustive generating models for all classes of regular and c-regular patterns. The generality of these algorithm models allows to easily experiment the exhaustive generation of pattern avoiding permutations. This is very useful to guess new classes of pattern avoiding permutations corresponding to classical sequences. We show an exhaustive list of well-known classes of regular and c-regular patterns together with their succession functions. These classes can be exhaustively generated in Constant Amortized Time (CAT for short) by our general models.

2 A Characterization for Right-Justified Permutation Patterns

In this section, after recalling some preliminaries on pattern avoiding permutations and right-justified patterns [11,13], we show a criterion for determining whether a pattern set is right-justified or not in Propositions 1 and 2.

Let S_n be the set of permutations on $\{1, 2, \ldots, n\}$. For the sake of simplicity, we use the one-line notation to represent a permutation. For that a permutation α will be written as a sequence $\alpha = a_1, a_2, \ldots, a_n \in S_n$, where a_i is the image of element i of α. Let $s = s_1, s_2, \cdots, s_k$ be a sequence of k pairwise distinct positive integers. The reduction of s, denoted by $red(s)$, is the permutation $\alpha = a_1, a_2, \cdots a_k$ of S_k such that $a_i < a_j$ whenever $s_i < s_j$. For example, $red(63594) = 41352$.

Given a permutation $\tau \in S_k$. A permutation $\alpha \in S_n$ is called *containing* τ, and τ is called a *pattern*, if we can extract a k-element subsequence s of α such that $red(s) = \tau$. Otherwise, we say that α *avoids* τ.

Example: The permutation $85217634 \in S_8$ contains pattern 312 as its subsequence $8, 1, 3$ satisfies that $red(8, 1, 3) = 312$. Whereas, 43768521 avoids 312.

Let P be a set of patterns. Denote

$$S_n(P) = \cap_{\tau \in P}\{\alpha \in S_n : \alpha \text{ avoids } \tau\},$$

the set of permutations of length n avoiding all patterns of P, and

$$S(P) = \cup_{n=1}^{\infty} S_n(P),$$

the set of all permutations avoiding P.

In the literature, the enumeration for permutations avoiding a pattern or a set of patterns received much attention. Some popular enumerations for $S_n(P)$ are listed together with their detailed references in Tables 1, 2 and 3. From that we can see that many classical sequences like Fibonacci, Binary strings, Lucas numbers, etc. are variously counted as permutations avoiding different pattern sets. Moreover, when investigating pattern avoiding permutations one often considers pattern sets whose lengths are small. As seen in the tables, almost all considered patterns have length 3 or 4.

We also perceive that each element of $S_n(P)$ can be obtained from one in $S_{n-1}(P)$ by inserting n into one of its slots. Therefore, permutations avoiding P can be represented on a rooted generating tree whose nth level contains all elements of $S_n(P)$. For pattern sets P possessing some good property like right-justified or regular pattern, generating permutations avoiding P is more effective in the sense that the insertion operation in the generating tree can be replaced by the one-position transition to the left on each level.

Next, we recall the definition of right-justified patterns which is an essential factor for the regular pattern definition in [11,13]. Note that many patterns considered in the literature are regular although showing their right-justifiedness was often mentioned weakly or even omitted. In this manuscript, we would like to give a simple criterion for recognizing right-justified patterns. Using this characterization, we will give two efficient algorithms for generating permutations avoiding (c-)regular patterns in Sects. 3 and 4.

Let $\alpha \in S_n$. When the entry n is not at the rightmost (resp. leftmost) position, denote $\alpha^{\rightarrow}$ (resp. $\alpha^{\leftarrow}$) the permutation obtained from α by moving the highest entry n of α to the right (resp. left) one position. For instance, $3142^{\rightarrow} = 3124$, $3142^{\leftarrow} = 3412$, and there does not exist $3124^{\rightarrow}$ or $4312^{\leftarrow}$.

Definition 1. *Given a pattern set P. We say that P is* right-justified *if for any permutation α avoiding P, we can move n to the right (if it is possible) and still have a permutation $\alpha^{\rightarrow}$ which avoids P.*

Example:

(i) $P = \{132\}$ is not right-justified since we have $3412 \in S_4(132)$ but $3142 \notin S_4(132)$, where 3142 is obtained from 3412 by moving 4 to the right. Moreover, it is easily seen that $P = \{312\}$ is right-justified by considering the highest element and element swapped in each permutation avoiding 312. More generally, a pattern set P containing a unique permutation $a_1, a_2, \cdots, a_k \in S_k$ is a right-justified pattern if and only if $a_1 = k$.

(ii) $P = \{312, 123\}$ is not right-justified since $132 \notin S_3(P)$ but moving 3 to its right we get 123 which in fact contains 123 of P. Later, we will see that showing $P = \{312, 132\}$ is right-justified is easy using the right-justified characterization.

In order to do that, we first consider P whose elements are of the same length. We have the following

Proposition 1. *Given a positive integer k. Let P be a pattern set such that every element of P is of length k. Then P is right-justified if and only if for each $\tau \in P$ we have $\tau^{\leftarrow} \in P$. In other words, for any permutation of P we can move k to the left one position and then have a permutation in P.*

Proof. Assume that P is right-justified and $\tau \in P$. On the contrary, we suppose that $\tau^{\leftarrow} \notin P$. Consider all permutations of length k of $S(P)$. By the Definition 1, since $\tau^{\leftarrow} \notin S_k(P)$, we have $\tau = (\tau^{\leftarrow})^{\rightarrow} \notin S(P)$ which contradicts the hypothesis that $\tau \in P$.

Conversely, let $\alpha \notin S_n(P)$. Assume that $\alpha_i = n$. We consider a sub-sequence u of $\tau^{\leftarrow}$.

$$u = a'_{i_1}, a'_{i_2}, \ldots, a'_{i_\ell}$$

- If $\ell \neq k$, then since all permutations in P have the same length, we must have then $red(u) \notin P$.
- If $\ell = k$ and if $\{i, i+1\}$ is not a subset of $\{i_1, i_2, \ldots, i_\ell\}$, then u is also a k-subsequence of α. Since α avoids P, then $red(u) \notin P$.
- If $\ell = k$ and if $\{i, i+1\}$ is a subset of $\{i_1, i_2, \ldots i_\ell\}$, then

  $$u = a'_{i_1}, a'_{i_2}, \ldots, a_{i+1}, n \ldots a'_{i_k}$$

  and

  $$red(u) = b_1, b_2, \ldots, b_p, k, \ldots b_k$$

  Consider the sequence v obtained from u by moving element n to the left one position. In fact,

  $$v = a_{i_1} \ldots a_{i_p}, n, a_{i+1}, \ldots, a_{i_k}.$$

  It is straightforward that v is a k-subsequence of α and $red(v) = red(u)^{\leftarrow}$. Since $\alpha \notin S(P)$ and since all permutations of P have the length k, we have $red(v) \notin P$. By the hypothesis, $red(u) = red(v)^{\rightarrow} \notin P$.

Therefore, in any cases, $red(u) \notin P$, and hence $\alpha^{\rightarrow} \notin S_n(P)$. This completes the proof. □

Now, we consider pattern sets P whose patterns may have different lengths. Actually, in this case, to check whether P is right-justified or not looks more complicated. We have to decompose P in subsets P_i where all permutations in P_i have the same lengths. Let $k_1 < k_2 < \cdots < k_p$ be the different lengths of the permutations in P and let P_i be the subset of P containing all the permutations with length k_i of P. Then

$$P = P_1 \cup P_2 \cup \cdots \cup P_p$$

We have the following

Proposition 2. *Let P be a pattern set. Then, P is right-justified if and only if for each $\tau \in P_k$, for $k = 1, 2, \ldots, p$, we have that $\tau^{\leftarrow}$ does not avoid $\cup_{i \leq k} P_i$. In other words, for any permutations of P, we can move its highest entry to the left one position and still have a permutation not avoiding P.*

Proof. Assume that P is right-justified. Let $\tau \in P$. On the contrary, suppose that $\tau^{\leftarrow}$ avoids P. Since P is right-justified, we have $\tau = (\tau^{\leftarrow})^{\rightarrow}$ avoids P which is a contradiction.

Conversely, let $\alpha \in S_n(P)$ such that $a_i = n$. We need to prove that $\alpha^{\rightarrow} \in S_n(P)$. On the contrary, let u be a subsequence with minimum length of $\alpha^{\rightarrow}$ such that $red(u) = \tau \in P$ and τ has length k.

$$u = a'_{t_1}, a'_{t_2}, \ldots, a'_{t_\ell}.$$

It is straightforward that $\{i, i+1\}$ is a subset of $\{t_1, t_2, \ldots, t_\ell\}$, or otherwise u is also a sub-sequence of α, and that α does not avoid P. Hence,

$$u = a'_{t_1}, a'_{t_2}, \ldots, a_{i+1}, n, \ldots, a'_{t_\ell},$$

and

$$red(u) = b_1, b_2, \ldots, b_p, k, \ldots, b_k \in P.$$

Since P satisfies the sufficient conditions in the Proposition,

$$red(u)^{\leftarrow} = b_1, b_2, \ldots, k, b_p, \ldots, b_k$$

contains a pattern with length $\leq k$ of P. Consider the sequence v obtained from u by interchange entry n and a_{i+1}. It is easily seen that v is a subsequence of α and $red(v) = red(u)^{\leftarrow}$. This gives that α contains a pattern of P which is a contradiction. □

As a sequence, the following patterns are right-justified. Note that these patterns were investigated in many contexts of pattern avoiding permutations, generating trees in the literature.

- $P = \{321, 231\}$, where $S_n(P)$ counts the sequence $\{2^{n-1}\}_{n\geq 1}$;
- $P = \{321, 3412, 4123\}$, where $S_n(P)$ counts Pell numbers;
- $P = \{321, 3412\}$, where $S_n(P)$ counts even index Fibonacci numbers;
- $P = \{2134, 2143, 2413, 4213\}$, where $S_n(P)$ counts central binomial coefficients $\binom{2n-2}{n-1}$;
- $P = \{321, 312, 23\ldots(p+1)1\}$, where $S_n(P)$ counts p-generalized Fibonacci numbers.

It is noted that for two permutations τ and α, deciding whether τ is a pattern of α is an NP-complete problem [17]. The following conjecture is for the NP-completeness of the right-justified checking problem.

Conjecture 1. *It is an NP-complete problem to check whether a pattern set is right-justified or not.*

However, for a pattern set P in which every pattern has the same length k, checking its right-justifiedness can be done in $\mathcal{O}(km \log m)$ by the following simple algorithm, where m is the number of patterns in P.

Algorithm 1. Algorithm for checking a pattern set P, in which every pattern has the same length k, is right-justified or not.

```
BOOL Check_Right_Justified(P, m, k)
⟨ Sort P in the lexicographic order in O(km log m) ⟩
for i := 2 to m do
   α ← the i^th pattern in P
   α' ← the (i − 1)^th pattern in P
   if α^← = α' then
      return FALSE
   end if
end for
return TRUE
end procedure
```

3 Regular Patterns

In this section, after giving the definition of regular pattern we will show the succession function for the permutation class avoiding a pattern set.

For $\alpha \in S_n$, we denote $\alpha^{\downarrow i}$ the permutation obtained from α by inserting $(n+1)$ into position i from the right of α. Now let P be a permutation pattern set. A site i of a permutation $\alpha \in S_n(P)$ is called an *active site* associated to P if $\alpha^{\downarrow i} \in S_{n+1}(P)$. In that case we say that $\alpha^{\downarrow i}$ is a successor of α. Notice that for right-justified patterns, active sites of a permutation avoiding it are numbered consecutively from 1. Let $\chi_P(i, \alpha)$ be the number of active sites of $\alpha^{\downarrow i}$. We have the following definition

Definition 2 (Regular pattern). *Let P be a set of permutation patterns. P is called* regular *if it satisfies the following properties:*

- *permutation* $1 \in S_1(P)$ *has two successors;*
- P *is right-justified;*
- *for any* $n \geq 1$, *and* $\alpha \in S_n(P)$, $\chi_P(i, \alpha)$ *does not depend on* α *but solely on* i *and the number* k *of active sites of* α. *In this case, we denote* $\chi_P(i, \alpha)$ *by* $\chi_P(i, k)$ *and we call it* succession function.

Let P be a regular set of patterns characterized by its succession function χ. The set of productions

$$(k) \rightsquigarrow (\chi_P(1, k)) \dots (\chi_P(k, k)),$$

for $k \geq 1$, is called the succession rule corresponding to P. For instance, Fig. 1 shows the generating tree for permutations avoiding pattern 321 and Fig. 2 shows the same generating tree where each node is labeled by the number of active sites of its corresponding nodes. We can see that, in Fig. 2, for instance, node (3) at the level 2 illustrates that the corresponding permutation 12 in Fig. 1 has two active sites. Moreover, every node labeled by (3) in Fig. 2 will generate nodes labeled sequentially by (4), (2), and (3).

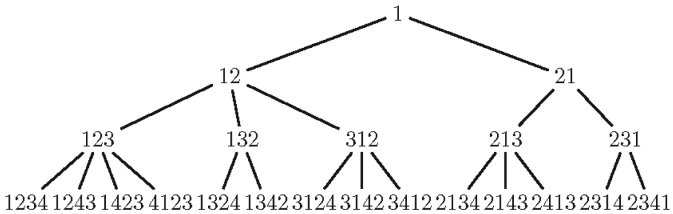


Fig. 1. The first four levels of the generating tree corresponding to $\chi(i, k) = i$ for $i = 2, 3, \cdots, k$ and $\chi(i, k) = k + 1$ for $i = 1$ producing $S(321)$ permutations in Table 1. The generating tree coincides with the computation tree of the generating procedure.

The succession functions for some well-known classes of regular patterns are given in Tables 1 and 2.

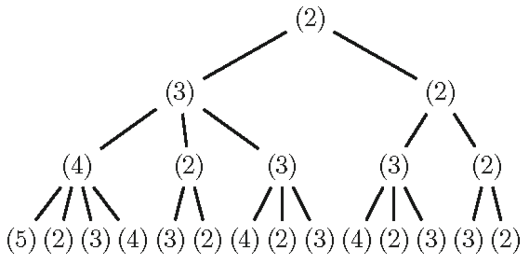


Fig. 2. The first four levels of the generating tree illustrating the number of active sites for each node of the tree in Fig. 1

Table 1. Several classes of regular patterns given by counting sequences together with the $\chi(i,k)$ function.

Class	P	$\chi_P(i,k)$
2^{n-1}[3]	$\{321, 312\}$	2
	$\{321, 231\}$	$k+1$ if $i=1$ 1 *otherwise*
Pell numbers [3,16]	$\{321, 3412, 4123\}$	3 if $i=1$ 2 *otherwise*
	$\{312, 4321, 3421\}$	3 if $i=2$ 2 *otherwise*
Even index Fibonacci numbers [3,16]	$\{321, 3412\}$	$k+1$ if $i=1$ 2 *otherwise*
	$\{321, 4123\}$	3 if $i=1$ i *otherwise*
	$\{312, 4321\}$	3 if $k=3$ and $i=3$ $i+1$ *otherwise*
Catalan numbers [20]	$\{312\}$	$i+1$
	$\{321\}$	$k+1$ if $i=1$ i *otherwise*
Schröder numbers [15]	$\{1234, 2134\}$	$k+1$ if $i=1$ or $i=2$ i *otherwise*
	$\{1324, 2314\}$	$k+1$ if $i=1$ or $i=k$ $i+1$ *otherwise*
	$\{4123, 4213\}$	$k+1$ if $i=k$ $i+2$ *otherwise*
Grand Dyck [16]	$\{1234, 1324, 2134, 2314\}$	$k+1$ if $i=1$ 3 if $i=2$ i *otherwise*
	$\{1324, 2314, 3124, 3214\}$	3 if $i=1$ $i+1$ *otherwise*
Fibonacci numbers [6]	$\{321, 312, 231\}$	1 if $i=2$ 2 *otherwise*

4 *c*-Regular patterns

Recall that not all right-justified patterns are regular. Actually, there exist right-justified patterns where the numbers of active sites of successors of a node α with k active sites may vary according to the shape of α. In case for each shape of α, if we can get the same rule for the number of active sites of its successors, then exhaustive algorithms as well as generating trees still reveal interesting

Table 2. Several classes of regular patterns given by variable length patterns together with the $\chi(i,k)$ function. Notice that for each class of variable length pattern here, for some particular values of p or m, we retrieve some cases in the Tables 1 and 2. For example, the case $T = \{321, 231, (p+1)12\dots p\}$, if $p = 2$, then we retrieve the case $T = \{321, 231, 321\}$ in the Table 1; and $p = \infty$ corresponds to the case $T = \{321, 231\}$ above.

Class	P	$\chi_P(i,k)$
A pattern of length 3 and a variable length pattern	$\{321, (p+1)12\dots p\}$ [3,8]	$k+1$ *if* $i=1$ and $k<p$ p if $i=1$ and $k=p$ i *otherwise*
	$\{321, p(p+1)12\dots(p-1)\}$ [3,8]	$k+1$ if $i=1$ i if $1<i<p-1$ $p-1$ *otherwise*
	$\{312, (p+1)p\dots 21\}$ [8]	p if $k=p$ and $i=p$ $i+1$ *otherwise*
A pattern of length 3, a pattern of length 4 and a variable length pattern	$\{321, 3412, (p+1)12\dots p\}$ [3]	$k+1$ if $i=1$ and $k<p$ p if $i=1$ and $k=p$ 2 *otherwise*
Generalized Fibonacci numbers	$\{321, 231, (p+1)12\dots p\}$ [3]	$k+1$ if $i=1$ and $k<p$ k if $i=1$ and $k=p$ 1 *otherwise*
A pattern of length 3 and two variable length pattern	$\{321, p(p+1)12\dots(p-1)$, $(p+1)12\dots p\}$ [3]	$k+1$ if $i=1$ and $k<p$ p if $i=1$ and $k=p$ $p-1$ if $i=p$ and $k=p$ i *otherwise*

properties. This section is to investigate such patterns, which are called c-regular patterns.

Let $\alpha \in S_n(P)$. We associate to α a non-zero integer, called its color. If the numbers of active sites of its successors does not depend on α but only the number of its active sites, we say that α has no color or the color of α is 0. We extend the previous function χ such that it transforms a triple (i,α,c) into a couple $(\mu(i,\alpha,c),\nu(i,\alpha,c))$ satisfying that if $\alpha \in S_n(P)$ has the color c then $\alpha^{\downarrow i} \in S_{n+1}(P)$ has μ active sites and color ν. Precisely, we have the following definition.

Definition 3 (**c-regular pattern**). *A set of patterns P is called* colored regular *(c-regular for short) if*

- *permutation $1 \in S_1(P)$ has two successors;*
- *P is right-justified;*

– *for any* $n \geq 1$ *and* $\alpha \in S_n(P)$, $\chi_P(i, \alpha, c)$ *does not depend on* α *but only on* i, *on* c *and on the number* k *of active sites of* α. *In this case we denote* $\chi_P(i, \alpha, c)$ *by* $\chi_P(i, k, c)$ *generalizes the succession function* $\chi(i, k)$.

Let P be a regular set of patterns characterized by its succession function $\chi_P(i, k, c) = (\mu(i, k, c), \nu(i, k, c))$. The set of productions

$$(k_c) \rightsquigarrow (\mu(1, k, c)_{\nu(1,k,c)}) \dots (\mu(k, k, c)_{\nu(k,k,c)}),$$

is called *colored succession rule* which corresponds to the set P of patterns. See Fig. 3 for an example. In this figure, both permutations 21 and 123 have 2 active sites, but they have different colors. Hence, we can see that they will generate nodes whose number of active sites are different. Moreover, permutations 21 and 132 have the same number of active sites and colors. Both of them are labeled by (2_1) and they both generate nodes labeled (1) and 2_0. It is also noted that in this case, pattern set $\{321, 312, 2341\}$ is c-regular but it is not regular.

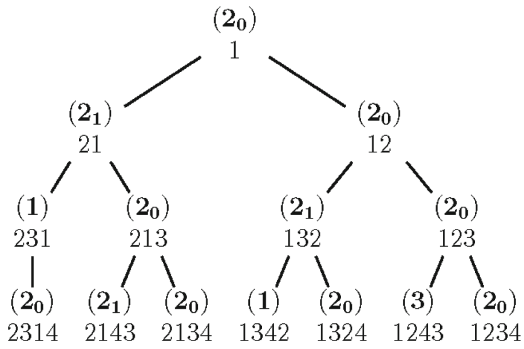


Fig. 3. The first four levels of the generating tree corresponding the class of generalize Fibonacci numbers with its succession rule described in Table 3 producing $S(321, 312, 2341)$ permutations. The generating tree coincides with the computation tree of the generating procedure.

The succession functions for some well-known classes of c-regular patterns are given in Table 3.

5 Constant Amortized Time Algorithms

In this section, we introduce two general algorithm models to generate all permutations avoiding a given regular or c-regular pattern. Algorithm 2 is first mentioned by Duckes et al. in [13], but they do not show the efficacy of the algorithm for the generated objects. Here we prove the 'CAT-ity' of our generating algorithms for all classes given in the tables.

Table 3. Several classes of c-regular patterns given by counting sequences together with the $\chi(i,k,c) = (\mu(i,k,c), \nu(i,k,c))$ function.

Class	P	$\mu_P(i,k,c)$	$\nu_P(i,k,c)$
Grand Dyck	$\{2134, 2143, 2413, 4213\}$ [16]	0 if $i < k-p$ $k+1$ *otherwise*	$k+1-i$ if $i \geq k-p$ 0 *otherwise*
	$\{1234, 1243, 1423, 4123\}$ [16]	0 if $i < k-p$ $k+1$ *otherwise*	$p+1$ if $i = k$ $k-i$ if $k-p \leq i < k$ 0 *otherwise*
Generalized Fibonacci numbers	$\{321, 312, 23\ldots(p+1)1\}$ [6]	1 if $c = p-2$ and $i = 2$ 2 *otherwise*	0 if $i = 1$ or $c = p-2$ $c+1$ *otherwise*
A pattern of length 3, a pattern of length 4 and a variable length pattern	$\{321, 4123, 34\ldots(p+1)12\}$ [3]	2 if $i = 2$ or ($i = 3$ and $c = p-3$) 3 *otherwise*	$c+1$ if $i = 3$ and $c < p-3$ 0 *otherwise*

Algorithm 2. Algorithm for generating permutations avoiding a regular pattern characterized by the succession function $\chi(i,k)$. The first call is Gen_Avoid$(1,2)$.

```
procedure Gen_Avoid(size, k)
local i
if size = n then
   Print(α)
else
   size := size + 1
   α := [α, size]        (* insert the value size to the end of α *)
   Gen_Avoid(size, χ(1,k))
   for i := 2 to k do
      α := α · (size − i + 2, size − i + 1)
      Gen_Avoid(size, χ(i,k))
   end for
   for i := k downto 2 do
      α := α · (size − i + 2, size − i + 1)
   end for
end if
end procedure
```

Let $\alpha \in S_n(P)$. We have $\alpha^{\downarrow i}$ and $\alpha^{\downarrow(i+1)}$ differ by a transposition. More precisely, $\alpha^{\downarrow i} = \alpha^{\downarrow(i+1)} \cdot (n-i+2, n-i+1)$. Generally, the insertion of an element in a list is not an efficient operation but the transposition of two element is an efficient elementary operation. Those are crucial in the efficaciousness of our generating algorithms. Further algorithm details are shown in Algorithms 2 and 3.

Lemma 1 ([19]). *If a recursive generating procedure satisfies the following three properties:*

Algorithm 3. Algorithm for generating permutations avoiding a c-regular pattern characterized by the succession function $\chi(i,k,c)$. The first call is Gen_Avoid$(1,2,0)$. The initial permutation is 1. In case when only one color is allowed, Gen_Avoid$(size,k,c)$ coincides with Gen_Avoid$(size,k)$.

```
procedure Gen_Avoid(size, k, c)
local i, u, v
if size = n then
   Print(α)
else
   size := size + 1
   α := [α, size]
   (u, v) := χ(1, k, c)
   Gen_Avoid(size, u, v)
   for i := 2 to k do
      α := α · (size − i + 2, size − i + 1)
      (u, v) := χ(i, k, c)
      Gen_Avoid(size, u, v)
   end for
   for i := k downto 2 do
      α := α · (size − i + 2, size − i + 1)
   end for
end if
end procedure
```

(i) the computation amount of a given call is proportional to its degree, disregarding the recursive calls;
(ii) each call has degree zero or at least two;
(iii) at the completion of each recursive call a new word is generated,

then the generating procedure is CAT.

The CAT property of the generating algorithm for Fibonacci family has already proved in [6]. Note that the number of recursive calls is the value of succession function ($\chi(i,k)$ or $\mu(i,k,c)$) and the number of operations in each Gen_Avoid$(size,k)$ and Gen_Avoid$(size,k,c)$ is negligible.

For the generating algorithm of the second succession function of the binary words in the first row of Table 1, considering the level n, we have:

$$\frac{\#recursive\ calls}{\#generated\ objects} = \frac{2^n + 2^{n-1} + \ldots + 2 + 1}{2^n} \simeq 2$$

satisfying the requirement of a CAT algorithm.

Apart from that, regarding all other succession functions $\chi(i,k)$, $\chi(i,k,c)$ in this paper, we perceive that all of them are at least 2 which satisfies Lemma 1.

6 Conclusion and Open Problems

We have characterized the right-justified property of regular patterns. This allows to build recursive generating algorithms for given classes of regular

patterns avoiding permutations, which has the CAT property. By using the succession function method and its induced generating algorithms, we may easily guess some new classes of pattern avoiding permutations. Usually, the proofs of new pattern avoiding permutation classes are only technical issues and take too long space, so let we show them in a long version of the paper.

Besides, by changing the values of the parameters i, j, k, we can obtain new succession functions generating objects with the same cardinality as the considered class. However, the corresponding pattern avoiding permutations is not easy to obtain. For example, for Pell numbers, although changing its succession function to:

$$\chi_P(i,k) = \begin{cases} 3 \text{ if } i = k, \\ 2 \text{ otherwise}, \end{cases}$$

generates Pell numbers, we have not found yet any corresponding class of pattern avoiding permutations. It would be more interesting if we can find a changing parameters way from existing succession functions to explore new combinatorial object classes.

Acknowledgment. This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2016.05. The second author is partially funded by the PSSG No.6.

References

1. Bacchelli, S., Barcucci, E., Grazzini, E., Pergola, E.: Exhaustive generation of combinatorial objects by ECO. Acta Informatica **40**, 585–602 (2004)
2. Banderier, C., Flajolet, P., Gardy, D., Bousquet-Melou, M., Denise, A., Gouyou-Beauchamps, D.: Generating functions for generating trees. Discrete Math. **246**, 29–55 (2002)
3. Barcucci, E., Bernini, A., Poneti, M.: From Fibonacci to Catalan permutations. Pu.M.A. **17**(1–2), 1–17 (2006)
4. Barcucci, E., Del Lungo, A., Pergola, E., Pinzani, R.: ECO: a methodology for the enumeration of combinatorial objects. J. Difference Equ. Appl. **5**, 435–490 (1999)
5. Barcucci, E., Vajnovszki, V.: Generalized Schröder permutations. Theoret. Comput. Sci. **502**, 210–216 (2013)
6. Baril, J.L., Do, P.T.: ECO-generation for p-generalized fibonacci and lucas permutations. Pu.M.A. **17**, 1–19 (2006)
7. Brlek, S., Duchi, E., Pergola, E., Rinaldi, S.: On the equivalence problem for succession rules. Discrete Math. **298**, 142–154 (2005)
8. Chow, T., West, J.: Forbidden sequences and Chebyshev polynomials. Discrete Math. **204**, 119–128 (1999)
9. Del Lungo, A., Frosini, A., Rinaldi, S.: ECO method and the exhaustive generation of convex polyominoes. In: Calude, C.S., Dinneen, M.J., Vajnovszki, V. (eds.) DMTCS 2003. LNCS, vol. 2731, pp. 129–140. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45066-1_10
10. Deutsch, E., Ferrari, L., Rinaldi, S.: Production matrices. Adv. Appl. Math. **34**, 101–122 (2005)
11. Do, P.T.: Arbres de génération et génération exhausitive. Ph.D. Thesis, University of Burgundy, France (2008)

12. Do, P.T., Rossin, D., Tran, T.T.H.: Permutations weakly avoiding barred patterns and combinatorial bijections to generalized Dyck and Motzkin paths. Discrete Math. **320**, 40–50 (2014)
13. Dukes, W.M.B., Flanagan, M.F., Mansour, T., Vajnovszki, V.: Combinatorial gray codes for classes of pattern avoiding permutations. Theoret. Comput. Sci. **396**, 35–49 (2008)
14. Ferrari, L., Pergola, E., Pinzani, R., Rinaldi, S.: Jumping succession rules and their generating functions. Discrete Math. **271**, 29–50 (2003)
15. Gire, S.: Arbres, permutations à motifs exclus et cartes planaires: quelques problèmes algorithmiques et combinatoires. Ph.D. Thesis, Université Bordeaux 1 (1993)
16. Guibert, O.: Combinatoire des permutations à motifs exclus en liaison avec mots, cartes planaires et tableaux de Young. Ph.D. Thesis, Université Bordeaux 1 (1995)
17. Buss, J.F., Bose, P., Lubiw, A.: Pattern matching for permutations. In: Proceedings of the 6th Workshop on Algorithms and Data Structures (1993)
18. Sabri, A., Vajnovszki, V.: More restricted growth functions: Gray codes and exhaustive generations. To appear in Graphs and Combinatorics (2017)
19. Vajnovszki, V.: Gray visiting Motzkins. Acta Informatica **38**, 793–811 (2002)
20. West, J.: Generating trees and the catalan and Schröder numbers. Discrete Math. **146**, 247–262 (1994)

Experimental Study of the Ehrhart Interpolation Polytope

Vissarion Fisikopoulos[1(✉)] and Zafeirakis Zafeirakopoulos[2(✉)]

[1] Oracle Corp., Neo Psychiko, Greece
vissarion.fisikopoulos@oracle.com
[2] Institute of Information Technologies, Gebze Technical University, Gebze, Turkey
zafeirakopoulos@gtu.edu.tr

Abstract. In this paper we define a family of polytopes called Ehrhart Interpolation Polytopes with respect to a given polytope and a parameter corresponding to the dilation of the polytope. We experimentally study the behavior of the number of lattice points in each member of the family, looking for a member with a single lattice point. That single lattice point is the h* vector of the given polytope. Our study is motivated by efficient algorithms for lattice point enumeration.

1 Introduction

A fundamental problem in discrete and computational geometry is to efficiently count or enumerate the lattice points of a polytope. Let $P \subseteq \mathbb{R}^d$ be a full dimensional polytope and L a lattice in $\mathbb{R}^n$. For any positive integer t, let $tP = \{tp : p \in P\}$ be the t-fold dilation of P and $\mathcal{L}_P(t) = \#(tP \cap L)$ be the counting function for the number of lattice points contained in tP. Now, let P be a lattice polytope, i.e., the vertices of P are lattice points. It is known, due to Ehrhart [2], that there exist rational numbers $a_0, \ldots, a_d$ such that $\mathcal{L}_P(t) = a_d t^d + a_{d-1} t^{d-1} + \cdots + a_1 t + a_0$. $\mathcal{L}_P(t)$ is called the Ehrhart polynomial of P. Note that the degree of $\mathcal{L}_P(t)$ is equal to the dimension of polytope P. Instead of considering $\mathcal{L}_P(t)$ in the monomial basis, we can also view it as a polynomial in the binomial basis $\left\{\binom{t+d-i}{d}\right\}_{i=0,\ldots,d}$ of polynomials of degree up to d. Then from [6], we have that

$$\mathcal{L}_P(t) = \sum_{j=0}^{d} h_j^* \binom{t+d-j}{d} \text{ and } h_j^* \in \mathbb{N} \tag{1}$$

where $(h_0^*, h_1^*, \ldots, h_d^*)$ is the h^* vector of P.

The h^* vector was the subject of many studies in the last decades [1,6,8]. A lot of interesting results exist, but we will only mention the ones relevant for the definition of the Ehrhart Interpolation Polytope (see Sect. 2).

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 320–324, 2017.
https://doi.org/10.1007/978-3-319-72453-9_26

2 The Ehrhart Interpolation Polytope

In this section, we define the main object of our study, the Ehrhart Interpolation Polytope. For this we will use the H^* polyhedron. The main idea is that there is a number of known linear inequalities for h^* vectors, thus defining a polyhedron. This polyhedron is contained in the non-negative orthant of $\mathbb{R}^{d+1}$, since h^* vectors are non-negative.

Definition 1 *(H^* **polyhedron**). Given $d \in \mathbb{N}^*$, let*

$$H^* = \left\{ \begin{array}{lll} x \in \mathbb{R}^{d+1} : & & \\ x_0 = 1 & & [2] \\ x_i \geq 0 & ,0 \leq i \leq d & [6] \\ x_i \geq x_1 & ,2 \leq i < d & [1] \\ x_d + x_{d-1} + \cdots + x_{d-i} \leq x_0 + x_1 + \cdots + x_{i+1} & ,0 \leq i \leq \lfloor \frac{d-1}{2} \rfloor & [1] \\ x_0 + x_1 + \cdots + x_i \leq x_d + x_{d-1} + \cdots + x_{d-i} & ,0 \leq i \leq \lfloor \frac{d-1}{2} \rfloor & [1] \\ x_1 + x_2 + \cdots + x_i \leq x_{d-1} + x_{d-2} + \cdots + x_{d-i} & ,0 \leq i \leq \lfloor \frac{d-1}{2} \rfloor & [8] \\ x_{d-1} + x_{d-2} + \cdots + x_{d-i} \leq x_2 + x_3 + \cdots + x_{i+1} & ,0 \leq i \leq \lfloor \frac{d-1}{2} \rfloor & [8] \end{array} \right\}$$

Lemma 1. *All h^* vectors of d-dimensional polytopes with at least one interior lattice point are lattice points in H^*.*

For special cases of d it is possible to have more constraints that could yield a more refined H^*.

The polyhedron H^* depends only on the dimension d. If we are given a polytope $P \subseteq \mathbb{R}^d$, we can obtain upper bounds for the h^* vector of P. In [7], Stanley proves the following monotonicity theorem.

Theorem 1 **([7]).** *If P and Q are polytopes in $\mathbb{R}^n$ and $P \subseteq Q$, then $h^*_{P,i} \leq h^*_{Q,i}$, where h^*_P and h^*_Q are the h^* vectors of P and Q respectively.*

The above result could yield upper bounds for the h^* vector of a given polytope P by constructing the smallest hypercube C containing P. The h^* vectors of lattice hypercubes are easy to compute and this way we bound from above all coordinates of the h^* vector of P as $h^*_{P,i} \leq h^*_{C,i}$ for $0 \leq i \leq d$. We use the constraints coming from the bounding hypercube together with the one defined by Eq. 1 to define the Ehrhart Interpolation Polytope.

Definition 2 (Ehrhart Interpolation Polytope). *Given a polytope $P \subseteq \mathbb{R}^d$ with at least one lattice point in its interior and $t \in \mathbb{N}^*$, we define the Ehrhart Interpolation Polytope of P in dilation t*

$$\mathcal{E}_P(t) = \left\{ \begin{array}{l} x \in \mathbb{R}^{d+1} : \\ x \in H^*, \\ \sum_{i=0}^{d} x_i \binom{t+d-i}{d} = \mathcal{L}_P(t), \\ x_i \leq h^*_{C,i},\ 0 \leq i \leq d \end{array} \right\} \tag{2}$$

where $C \subseteq \mathbb{R}^d$ is the smallest cube containing P.

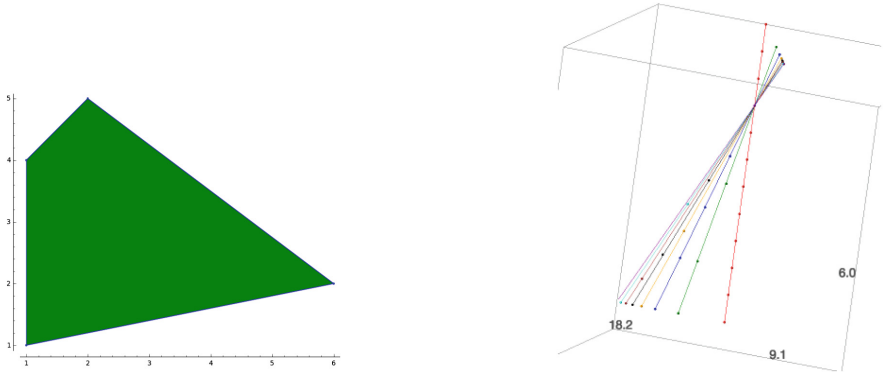


Fig. 1. The polytope P (left) and the associated $\mathcal{E}_P(t)$ (right) from Example 1.

Note that $\mathcal{E}_P(t)$ is indeed a bounded polyhedron, since it is contained in the intersection of the positive orthant with a hyperplane whose normal vector is a strictly positive vector (containing binomial coefficients). Moreover, observe that the h^* vector of P is contained in $\mathcal{E}_P(t)$ for all $t \in \mathbb{N}^*$ by construction, for all polytopes P containing at least one lattice point in their interior. Generically, H^* and $\mathcal{E}_P(t)$ will have dimension d and $d-1$ respectively.

Example 1. Let $P \subseteq \mathbb{R}^2$ be the convex hull of the points $(1,1),(1,4),(2,5),(6,2)$. Then the Ehrhart Interpolation Polytope $\mathcal{E}_P(t)$ is an 1-dimensional polytope in $\mathbb{R}^3$.

Figure 1 depicts P and $\mathcal{E}_P(t)$ for $t = 1, 2, \ldots, 8$. For $t = 1, 2, \ldots, 8$, there are $12, 3, 6, 2, 3, 2, 3, 1$ lattice points in each segment respectively. For $t = 8$ (purple), $\mathcal{E}_P(t)$ contains a single lattice point. This lattice point is the h^* vector of the polytope P. Note that for dilations greater than 8, it is possible to have more than one lattice points, see Sect. 3.

The single lattice point in $\mathcal{E}_P(8)$ is $(1, 12, 9)$. The binomial basis for polynomials in dimension 2 is $\left\{\frac{(t+1)(t+2)}{2}, \frac{t(t+1)}{2}, \frac{(t-1)t}{2}\right\}$. By evaluating Eq. 1 we get the Ehrhart polynomial of P which is $11t^2 + 3t + 1$.

3 Experiments and Statistics

We experimentally study the number of lattice points of the $\mathcal{E}_P(t)$ as a function of the dilation t. The goal is to find a dilation t such that $\mathcal{E}_P(t)$ contains a single lattice point. Then, that lattice point is the h^* vector of P. Our experiments indicate that as we increase the dilation, after some point, the h^* vector becomes a vertex of the integer hull of the Ehrhart Interpolation Polytope. Moreover, experimental evidence indicates that after a certain threshold, for some dilations the integer hull of the Ehrhart Interpolation Polytope is 0-dimensional, i.e., a single point. This can already be observed in the low dimensions; see Example 1.

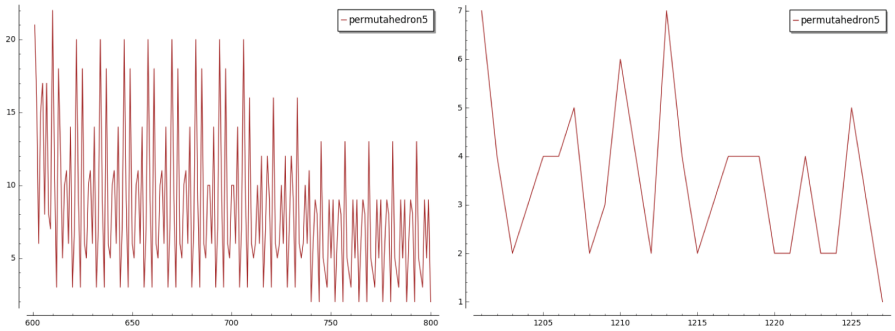


Fig. 2. The number of lattice points in the Ehrhart Interpolation Polytope for dilations 600–800 (left) and dilations 1200–1227 (right). Dilation 1227 contains one lattice point.

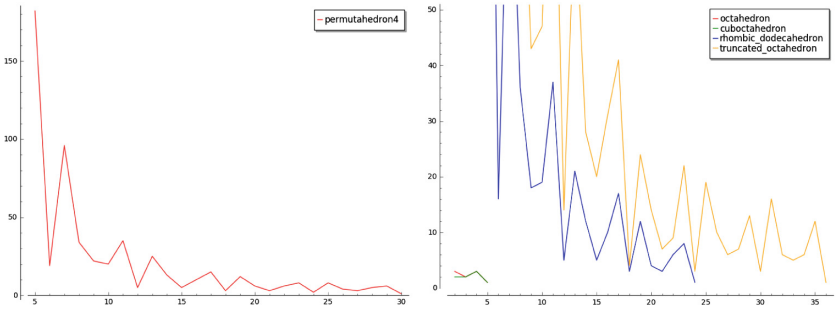


Fig. 3. Number of lattice points in the Ehrhart Interpolation Polytope for the permutahedron in dimension 4 (left) and for some solids (right) for dilations 5 to 30.

In this section we present some statistics about the number of lattice points in the Ehrhart Interpolation Polytope.

For our study we concentrate on cross polytopes, permutahedra, random simplices, and zonotopes in dimensions 3, 4, and 5, as well as Platonic solids that are lattice polytopes. For the exploration presented here we used the computer algebra system Sage [5].

We first focus on the permutahedron in dimension 5, to show some observations that hold for other families of polytopes as well. In Fig. 2, we see that the number of lattice points in successive dilations exhibits a periodic behavior. The lowest point during a period is related to the dimension of the polytope. Note that the permutahedron in dimension d is a $d-1$-dimensional polytope.

In Fig. 3, we can see that the Ehrhart Interpolation Polytope of the permutahedron in dimension 4 contains a single lattice point in dilation 30 on the left. On the right, we see the behavior of some Platonic Solids that are lattice polytopes.

Regarding timings, all above experiments have been performed on a personal computer in order of seconds.

4 Conclusion

The purpose of this short paper is to start a discussion on the study of Ehrhart Interpolation Polytopes. Our preliminary results indicate that it should be interesting to study more their combinatorial properties and provide rigorous experimental or analytical results.

Our original motivation for the definition of Ehrhart Interpolation Polytope was the computation of Ehrhart polynomials, using good approximations of the volume in large dilates of the polytope. Matthias Köppe [4] suggested to use Integer Linear Programming for finding the h^* vector in the Ehrhart Interpolation Polytope.

Interestingly, finding the Ehrhart polynomial reduces to a non-convex optimization problem, namely finding the minimum of the function counting the number of lattice points of a polytope varying dilation t. Finally, practical volume approximation algorithms [3] can be applied to yield bounds on the number of lattice points that could be used to refine $\mathcal{E}_P(t)$.

Acknowledgments. The second author acknowledges support from the project BAP 2016-A-27 of Gebze Technical University.

References

1. Beck, M., De Loera, J.A., Develin, M., Pfeifle, J., Stanley, R.P.: Coefficients and Roots of Ehrhart Polynomials. ArXiv Mathematics e-prints, February 2004
2. Ehrhart, E.: Sur la loi de réciprocité des polyèdres rationnels. C. R. Acad. Sci. Paris **26612**, 696–697 (1968)
3. Emiris, I.Z., Fisikopoulos, V.: Efficient random-walk methods for approximating polytope volume. In: 30th Annual Symposium on Computational Geometry, Kyoto, Japan, p. 318 (2014)
4. Köppe, M.: Personal communication
5. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 8.0.0) (2017). http://www.sagemath.org
6. Stanley, R.P.: Decompositions of rational convex polytopes. In: Srivastava, J., (ed.) Combinatorial Mathematics, Optimal Designs and Their Applications. Ann. Discrete Math. 6, 333–342. Elsevier (1980)
7. Stanley, R.P.: A monotonicity property of h-vectors and h*-vectors. Eur. J. Comb. **14**(3), 251–258 (1993)
8. Stapledon, A.: Additive number theory and inequalities in Ehrhart theory. Int. Math. Res. Not. **2016**(5), 1497–1540 (2015)

On Testing Isomorphism of Graphs of Bounded Eigenvalue Multiplicity

Takunari Miyazaki(✉)

Trinity College, Hartford, CT 06106-3100, USA
takunari.miyazaki@trincoll.edu

Abstract. Using Delsarte, Goethals and Seidel's fundamental theorem of spherical codes and designs, we investigate the complexity of graph-isomorphism testing. We derive a set of basic conditions under which testing isomorphism of graphs of bounded eigenvalue multiplicity is immediately reducible to testing isomorphism of graphs of bounded color multiplicity.

1 Introduction

Determining the computational complexity of testing isomorphism of finite algebraic and combinatorial structures is a long-standing unresolved problem. Since all such structures can be canonically represented by graphs, the *graph-isomorphism* problem has historically attracted the most attention.

By bounding graph parameters, a number of studies have demonstrated considerable success. In particular, two of the earliest and most fundamental work approached the problem by bounding graphs' *vertex-color multiplicity* and separately *eigenvalue multiplicity* (in [2,3], respectively). In this report, we will revisit these two classes of graphs.

While both classes of graphs admit polynomial-time solutions, the problem for graphs of bounded color multiplicity has been regarded "easier", at least informally, than that for graphs of bounded eigenvalue multiplicity. The former has long been known to be *fixed-parameter tractable* (i.e., the bounded parameter is *not* involved in the exponent of the complexity) by Babai's seminal "tower-of-groups" method (see [2,9]; cf. [1]), whereas, for the latter, no simple solution exists to achieve fixed-parameter tractability (cf. [8]). In [3], the tower-of-groups algorithm is in fact used as a critical subroutine to solve the latter problem; however, in the end, the bounded parameter is involved in the exponent of the complexity, thereby not achieving fixed-parameter tractability. Our overall objective is to investigate deeper the relationship between the two problems.

In spectral graph theory, the automorphism groups of graphs of n vertices are often studied as embeddings in the real orthogonal group $\mathrm{O}_n(\mathbf{R})$. In this work, we will take a look at natural geometries in $\mathbf{R}^n$ induced by such embeddings and apply a fundamental theorem of spherical codes and designs.

In their seminal work [7], Delsarte, Goethals and Seidel lay the foundations of what is now called the *Delsarte theory* of spherical codes and designs (cf. [4]).

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 325–329, 2017.
https://doi.org/10.1007/978-3-319-72453-9_27

Their starting point is a theorem that gives an upper bound for the cardinality of a finite *s-distance set* (in which there can only be s distinct distances between any two points) on the unit sphere in $\mathbf{R}^d$ as a function of s and d (see [7, Theorem 4.8]). In our investigation, it turns out that the orbits of the automorphism groups of graphs can naturally be mapped to such spherical sets. With the help of this theorem, we will derive a set of basic conditions under which testing isomorphism of graphs of bounded eigenvalue multiplicity is immediately reducible to testing isomorphism of graphs of bounded color multiplicity.

In the theory of permutation groups, *primitive groups* form central building blocks and thus have been the main subject of intense study. Of particular focus have been primitive groups of *low rank* (here, the rank means the number of orbits in the natural coördinate-wise action on the cartesian square of the underlying set). Indeed, our main result will assert that, given graphs of bounded eigenvalue multiplicity, if the automorphism groups act on partitioned subsets of vertices as primitive groups of low rank, then the subsets define vertex colorings of bounded color multiplicity. While a great deal has been known about primitive groups (see, e.g., [6]), our proof requires no such knowledge and is algebraic-combinatorial in nature based on the Delsarte–Goethals–Seidel theorem.

To formally state the main result, we begin with some definitions. Let $X = (V, E)$ be a graph of n vertices. For $V = \{v_1, \ldots, v_n\}$, consider an n-dimensional real inner-product space $\mathbf{R}^n$ with an orthonormal basis $\{e_{v_1}, \ldots, e_{v_n}\}$. An action of a group on V then induces an orthogonal action, called the permutation representation, on $\mathbf{R}^n$ by permuting the coördinates of the vectors of $\mathbf{R}^n$.

For a partition $V = C_1 \cup \cdots \cup C_s$ and an orthogonal direct-sum decomposition $\mathbf{R}^n = W_1 \oplus \cdots \oplus W_t$, we call $((C_1, \ldots, C_s), (W_1, \ldots, W_t))$ a *partition-decomposition* pair for X, provided that all C_i and W_j are invariant under $\mathrm{Aut}(X)$ for $i = 1, \ldots, s$ and $j = 1, \ldots, t$. We further call such a pair *stable* if it satisfies the following conditions (i)–(iii) for each of $i = 1, \ldots, s$ and $j = 1, \ldots, t$.

In what follows, $p^*_{W_j} : V \to W_j$ denotes an extension of the orthogonal projection $p_{W_j} : \mathbf{R}^n \to W_j$ defined by $p^*_{W_j}(v) := p_{W_j}(e_v)$ for $v \in V$, and $\sim_j$ denotes an equivalence relation on V defined by $v \sim_j v'$ if $p^*_{W_j}(v) = p^*_{W_j}(v')$ for $v, v' \in V$.

(i) All vectors in $p^*_{W_j}(C_i)$ have the same length.
(ii) All equivalence classes of $\sim_1 |_{C_i} \wedge \cdots \wedge \sim_j |_{C_i}$ have the same size.
(iii) $\langle p^*_{W_j}(C_i) \rangle = 0$ or W_j.

Such partition-decomposition pairs are easily obtainable from adjacency matrices via spectral decompositions by a "refinement" algorithm (see [3, §4]); thus, if X's eigenvalue multiplicity is at most d, then $\dim_{\mathbf{R}}(W_j) \leq d$ for $j = 1, \ldots, t$.

Theorem 1. *Let X be a graph of eigenvalue multiplicity at most d such that $\mathrm{Aut}(X)$ acts primitively on all its orbits. If X has a stable partition-decomposition pair $((C_1, \ldots, C_s), (W_1, \ldots, W_t))$ such that the rank of $\mathrm{Aut}(X)|_{C_i}$ is at most r for $i = 1, \ldots, s$, then the following hold.*

(i) $|C_i| \leq f(d, r) := \binom{d+r-2}{r-1} + \binom{d+r-3}{r-2}$ *for* $i = 1, \ldots, s$.
(ii) *Isomorphism of such graphs can be tested in $O(f(d, r)!^2 n^c)$ time for an absolute constant $c > 0$.*

Theorem 1(ii) asserts that testing isomorphism of such graphs is fixed-parameter tractable. It is a consequence of Theorem 1(i); that is, we regard $(C_1, \ldots, C_s)$ as a vertex coloring and appeal to the aforementioned tower-of-groups algorithm to test isomorphism of graphs of bounded color multiplicity (see [2,9]).

2 Permutation Representations and Projections

We begin with a brief summary of the basic facts we will need. For the foundations of this subject, we refer the reader to [5].

Let $X = (V, E)$ be a graph of n vertices defined by an $n \times n$ adjacency matrix A. Regarding A as a real symmetric matrix, consider its distinct eigenvalues $\lambda_1, \ldots, \lambda_t$, all of which are real, and corresponding eigenspaces $W_1, \ldots, W_t$. Recall then that $\mathbf{R}^n = W_1 \oplus \cdots \oplus W_t$, where $W_1, \ldots, W_t$ are mutually orthogonal to one another. If the multiplicity of λ_i is d_i, then $\dim_{\mathbf{R}}(W_i) = d_i$ for $i = 1, \ldots, t$; if X is a *graph of eigenvalue multiplicity at most* d, then it means that $d_i \leq d$ for $i = 1, \ldots, t$.

For $V = \{v_1, \ldots, v_n\}$, consider the symmetric group $\mathrm{Sym}(V)$ and its natural action on the set of all unordered pairs of vertices $\binom{V}{2}$. The *automorphism group of* X is defined by $\mathrm{Aut}(X) := \{g \in \mathrm{Sym}(V) : E^g = E\}$.

The symmetric group $\mathrm{Sym}(V)$ also acts on $\mathbf{R}^n$ by permuting the coördinates of the vectors of $\mathbf{R}^n$. It is defined by, for an orthonormal basis $\{e_{v_1}, \ldots, e_{v_n}\}$ of $\mathbf{R}^n$, the permutation representation $^* : \mathrm{Sym}(V) \to \mathrm{GL}_n(\mathbf{R})$ such that $e_{v_i}^{g^*} = e_{v_i^g}$ for $i = 1, \ldots, n$ and $g \in \mathrm{Sym}(V)$. That is, the image $\mathrm{Sym}(V)^*$, consisting of all $n \times n$ permutation matrices, is a subgroup of the real orthogonal group $\mathrm{O}_n(\mathbf{R})$. Under this representation, regarding A as a real $n \times n$ matrix, it is easy to see that $\mathrm{Aut}(X) = \{g \in \mathrm{Sym}(V) : g^{*-1} A g^*\}$, which in turn implies

$$\mathrm{Aut}(X) = \{g \in \mathrm{Sym}(V) : W_i^{g^*} = W_i \text{ for } i = 1, \ldots, t\}.$$

If X is a graph of eigenvalue multiplicity at most d, then, as $\mathrm{Aut}(X)$ must stabilize the eigenspaces, $\mathrm{Aut}(X)$ is embeddable in $\mathrm{O}_d(\mathbf{R}) \times \cdots \times \mathrm{O}_d(\mathbf{R})$.

By contrast, if X is a *graph of color multiplicity at most* k, then it means that X is equipped with a vertex coloring $\phi : V \to \{1, \ldots, s\}$ such that the number of vertices sharing the same color is at most k, i.e., X has a partition of vertices $V = C_1 \cup \cdots \cup C_s$ such that $|C_i| \leq k$ for $i = 1, \ldots, s$. In this case, the group of color-preserving automorphisms is embeddable in $S_k \times \cdots \times S_k$.

We next state two elementary lemmas concerning orthogonal projections (cf. [3]). As usual, for a subspace W of $\mathbf{R}^n$, $p_W : \mathbf{R}^n \to W$ denotes the orthogonal projection of $\mathbf{R}^n$ on W.

Lemma 2. *If W is a subspace of $\mathbf{R}^n$, and $g \in \mathrm{O}_n(\mathbf{R})$, then $W^g = W$ if and only if $p_W(v)^g = p_W(v^g)$ for all $v \in \mathbf{R}^n$.* □

Lemma 3. *Let $E = \{e_1, \ldots, e_n\}$ be an orthonormal basis of $\mathbf{R}^n$ and G be a subgroup of $\mathrm{O}_n(\mathbf{R})$ of permutation matrices acting on E. If F is a G-invariant subset of E, and W is a G-invariant subspace of $\mathbf{R}^n$, then the following hold.*

(i) *Let $\sim$ be an equivalence relation on F defined by $e \sim e'$ if $p_W(e) = p_W(e')$ for $e, e' \in F$. The equivalence classes $F/\sim$ form a G-invariant partition.*
(ii) *The action of G on $F/\sim$ is isomorphic to the action of G on $p_W(F)$.* □

3 Bounding Vertex Partitions

We will sketch the proof of the main theorem. It is based on a fundamental theorem of the so-called Delsarte theory of spherical codes and designs (cf. [4]).

We first recall from [7, Theorem 4.8].

Theorem 4 (Delsarte–Goethals–Seidel). *If S is a finite set of points on the unit sphere in $\mathbf{R}^d$, and $s := |\{(x, y) : x, y \in S \text{ and } x \neq y\}|$, then*

$$|S| \leq \binom{d+s-1}{s} + \binom{d+s-2}{s-1}$$

(where $(\ ,\ )$ denotes the usual inner product in $\mathbf{R}^d$). □

Its proof is based on basic properties of harmonic polynomials and spherical harmonics. This novel approach has subsequently been extended to many other problems in spherical codes and designs (cf. [4]).

Central to the proof of Theorem 1(i) is the following.

Proposition 5. *Let X be a graph such that $\mathrm{Aut}(X)$ acts primitively on all its orbits. For a stable partition-decomposition pair $((C_1, \ldots, C_s), (W_1, \ldots, W_t))$ for X, let r_i denote the rank of $\mathrm{Aut}(X)|_{C_i}$ for $i = 1, \ldots, s$ and $d_j := \dim_{\mathbf{R}}(W_j)$ for $j = 1, \ldots, t$. Then, for each C_i, there is W_j such that*

$$|C_i| \leq \binom{d_j + r_i - 2}{r_i - 1} + \binom{d_j + r_i - 3}{r_i - 2}.$$

Proof. First, recall from p. 2 the stability conditions (i)–(iii) of partition-decomposition pairs, in particular, for $j = 1, \ldots, t$, the extension $p^*_{W_j} : V \to W_j$ of the orthogonal projection $p_{W_j} : \mathbf{R}^n \to W_j$ and the equivalence relation $\sim_j$ defined by $v \sim_j v'$ if $p^*_{W_j}(v) = p^*_{W_j}(v')$ for $v, v \in V$. Throughout, write $G := \mathrm{Aut}(X)$.

First, consider arbitrary C_i and $\sim_j$. By Lemma 3(ii), $C_i/\sim_j$ is a G-invariant partition. Since G acts primitively on all its orbits, by the stability condition (ii) of partition-decomposition pairs, $|C_i/\sim_j| = |C_i|$ or 1.

Clearly, $e_{v_1}, \ldots, e_{v_n}$ are all distinct, so the equivalence relation $\sim_1 \wedge \cdots \wedge \sim_t$ is in fact the equality relation $=$. Thus, for each C_i, there exists $\sim_j$ such that $|C_i/\sim_j| = |C_i|$. From now on, we consider C_i and $\sim_j$ such that $|C_i/\sim_j| = |C_i|$.

If $|C_i| = 1$, then we are done. So, we assume that $|C_i| \geq 2$. Since $|p^*_{W_j}(C_i)| = |C_i/\sim_j| = |C_i|$, by the stability condition (iii) of partition-decomposition pairs, $\langle p^*_{W_j}(C_i)\rangle = W_j \neq 0$.

Now, let $^* : G \to \mathrm{GL}_n(\mathbf{R})$ denote the permutation representation of G on $\mathbf{R}^n$ and $S := p^*_{W_j}(C_i)$. By Lemma 3(ii), the G-action on C_i is isomorphic to the G^*-action on S. By the stability condition (i) of partition-decomposition pairs, all the

vectors in S have the same length and thus form a spherical set. The G^*-action on S is clearly orthogonal and thus preserves the inner-products of the vectors of S. Since the rank of $G|_{C_i}$ is r_i, the cardinality of the set $\{(x, y) : x, y \in S \text{ and } x \neq y\}$ is at most $r_i - 1$. Consequently, if $d_j := \dim_{\mathbf{R}}(W_j)$, then, by Theorem 4,

$$|C_i| = |S| \leq \binom{d_j + r_i - 2}{r_i - 1} + \binom{d_j + r_i - 3}{r_i - 2}$$

as required. □

Theorem 1(i) asserts that the graph X's vertex partition $(C_1, \ldots, C_s)$ defines a vertex coloring of color multiplicity at most k, making $\mathrm{Aut}(X)$ embeddable in $S_k \times \cdots \times S_k$, for $k = f(d, r) := \binom{d+r-2}{r-1} + \binom{d+r-3}{r-2}$. By the tower-of-groups algorithm, $\mathrm{Aut}(X)$ is then computable in $O(k!^2 n^c)$ time for an absolute constant $c > 0$ (see [2,9]). In general, to test isomorphism of two connected graphs, it suffices to compute the automorphism group of the union of the two. Thus, testing isomorphism of such graphs is also fixed-parameter tractable.

Acknowledgement. The author takes pleasure in acknowledging the hospitality of Nihon University while this research was being undertaken. He would like to especially thank Seinosuke Toda for hosting the author, bringing the author's attention to this topic and stimulating conversations. The author would also like to acknowledge Kazuhiro Yokoyama for insightful comments.

References

1. Arvind, V., Das, B., Köbler, J., Toda, S.: Colored hypergraph isomorphism is fixed parameter tractable. Algorithmica **71**, 120–138 (2015)
2. Babai, L.: Monte carlo algorithms in graph isomorphism testing. Technical Report 79–10, Département de mathématiques et de statistique, Université de Montréal, Montréal (1979)
3. Babai, L., Grigoryev, Yu, D., Mount, D.M.: Isomorphism of graphs with bounded eigenvalue multiplicity. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, San Francisco, 5–7 May 1982, pp. 310–324. ACM, New York (1982)
4. Bannai, E., Bannai, E.: A survey on spherical designs and algebraic combinatorics on spheres. Eur. J. Combin. **30**, 1392–1425 (2009)
5. Biggs, N.: Algebraic Graph Theory. Cambridge Mathematical Library. Cambridge University Press, Cambridge (1993)
6. Cameron, P.J.: Finite permutation groups and finite simple groups. Bull. Lond. Math. Soc. **13**, 1–22 (1981)
7. Delsarte, P., Goethals, J.M., Seidel, J.J.: Spherical codes and designs. Geom. Dedicata **6**, 363–388 (1977)
8. Evdokimov, S., Ponomarenko, I.: Isomorphism of coloured graphs with slowly increasing multiplicity of jordan blocks. Combinatorica **19**, 321–333 (1999)
9. Furst, M., Hopcroft, J., Luks, E.: A subexponential algorithm for trivalent graph isomorphism. Technical Report 80–426, Department of Computer Science, Cornell University, Ithaca, NY (1980)

Data Modeling and Analysis

A Simple Streaming Bit-Parallel Algorithm for Swap Pattern Matching

Václav Blažej(✉), Ondřej Suchý, and Tomáš Valla

Faculty of Information Technology,
Czech Technical University in Prague, Prague, Czech Republic
vaclav.blazej@fit.cvut.cz

Abstract. The pattern matching problem with swaps is to find all occurrences of a pattern in a text while allowing the pattern to swap adjacent symbols. The goal is to design fast matching algorithm that takes advantage of the bit parallelism of bitwise machine instructions and has only streaming access to the input. We introduce a new approach to solve this problem based on the graph theoretic model and compare its performance to previously known algorithms. We also show that an approach using deterministic finite automata cannot achieve similarly efficient algorithms. Furthermore, we describe a fatal flaw in some of the previously published algorithms based on the same model. Finally, we provide experimental evaluation of our algorithm on real-world data.

1 Introduction

In the *Pattern Matching problem with Swaps* (Swap Matching, for short), the goal is to find all occurrences of any *swap version* of a pattern P in a text T, where P and T are strings over an alphabet Σ of length p and t, respectively. By the swap version of a pattern P we mean a string of symbols created from P by swapping adjacent symbols while ensuring that each symbol is swapped at most once (see Sect. 2 for formal definitions). The solution of Swap Matching is a set of indices which represent where occurrences of P in T begin. Swap Matching is an intensively studied problem due to its use in practical applications such as text and music retrieval, data mining, network security and biological computing [5].

The swap of two consecutive symbols is one of the most typical typing errors. It also represent a simpler version of swaps that appear in nature. In particular, the phenomenon of swaps occurs in gene mutations and duplications such as in the region of human chromosome 5 that is implicated in the disease called spinal muscular Atrophy, a common recessive form of muscular dystrophy [15]. While the biological swaps occur at a gene level and have several additional constraints and characteristics, which make the problem much more difficult, they do serve as

V. Blažej—Supported by the SGS CTU project SGS17/209/OHK3/3T/18.
O. Suchý—Supported by grant 17-20065S of the Czech Science Foundation.
T. Valla—Supported by the Centre of Excellence – Inst. for Theor. Comp. Sci. 79 (project P202/12/G061 of the Czech Science Foundation.).

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 333–348, 2017.
https://doi.org/10.1007/978-3-319-72453-9_28

a convincing pointer to the theoretical study of swaps as a natural edit operation for the approximation metric [2]. Indeed Wagner and Lowrance [17] suggested to add the swap operation when considering the edit distance of two strings.

Swap Matching was introduced in 1995 as an open problem in non-standard string matching [16]. The first result was reported by Amir et al. [2] in 1997, who provided an $O(tp^{\frac{1}{3}} \log p)$-time solution for alphabets of size 2, while also showing that alphabets of size exceeding 2 can be reduced to size 2 with a little overhead. Amir et al. [4] also came up with solution with $O(t \log^2 p)$ time complexity for some very restrictive cases. Several years later Amir et al. [3] showed that Swap Matching can be solved by an algorithm for the overlap matching achieving the running time of $O(t \log p \log |\Sigma|)$. This algorithm as well as all the previous ones is based on fast Fourier transformation (FFT).

In 2008 Iliopoulos and Rahman [14] introduced a new graph theoretic approach to model the Swap Matching problem and came up with the first efficient solution to Swap Matching without using FFT (we show it to be incorrect). Their algorithm based on bit parallelism runs in $O((t + p) \log p)$ time if the pattern length is similar to the word-size of the target machine. One year later Cantone and Faro [8] presented the Cross Sampling algorithm solving Swap Matching in $O(t)$ time and $O(|\Sigma|)$ space, assuming that the pattern length is similar to the word-size in the target machine. In the same year Campanelli et al. [7] enhanced the Cross Sampling algorithm using notions from Backward directed acyclic word graph matching algorithm and named the new algorithm Backward Cross Sampling. This algorithm also assumes short pattern length. Although Backward Cross Sampling has $O(|\Sigma|)$ space and $O(tp)$ time complexity, which is worse than that of Cross Sampling, it improves the real-world performance.

In 2013 Faro [11] presented a new model to solve Swap Matching using reactive automata and also presented a new algorithm with $O(t)$ time complexity assuming short patterns. The same year Chedid [10] improved the dynamic programming solution by Cantone and Faro [8] which results in more intuitive algorithms. In 2014 a minor improvement by Fredriksson and Giaquinta [12] appeared, yielding slightly (at most factor $|\Sigma|$) better asymptotic time complexity (and also slightly worse space complexity) for special cases of patterns. The same year Ahmed et al. [1] took ideas of the algorithm by Iliopoulos and Rahman [14] and devised two algorithms named Smalgo-I and Smalgo-II which both run in $O(t)$ for short patterns, but bear the same error as the original algorithm.

Our Contribution. We design a simple algorithm which solves the Swap Matching problem. The goal is to design a streaming algorithm, which is given one symbol per each execution step until the end-of-input arrives, and thus does not need access to the whole input. This algorithm has $O(\lceil \frac{p}{w} \rceil (|\Sigma| + t) + p)$ time and $O(\lceil \frac{p}{w} \rceil |\Sigma|)$ space complexity where w is the word-size of the machine. We would like to stress that our solution, as based on the graph theoretic approach, does not use FFT. Therefore, it yields a much simpler non-recursive algorithm allowing bit parallelism and is not suffering from the disadvantages of the

convolution-based methods. While our algorithm matches the best asymptotic complexity bounds of the previous results [8,12] (up to a $|\Sigma|$ factor), we believe that its strength lies in the applications where the alphabet is small and the pattern length is at most the word-size, as it can be implemented using only $7 + |\Sigma|$ CPU registers and few machine instructions. This makes it practical for tasks like DNA sequences scanning. Also, as far as we know, our algorithm is currently the only known streaming algorithm for the swap matching problem.

We continue by proving that any deterministic finite automaton that solves Swap Matching has number of states exponential in the length of the pattern.

We also describe the Smalgo (swap matching algorithm) by Iliopoulos and Rahman [14] in detail. Unfortunately, we have discovered that Smalgo and derived algorithms contain a flaw which cause false positives to appear. We have prepared implementations of Smalgo-I, Cross Sampling, Backward Cross Sampling and our own algorithm, measured the running times and the rate of false positives for the Smalgo-I algorithm. All of the sources are available for download.[1]

This paper is organized as follows. First we introduce all the basic definitions, and also recall the graph theoretic model introduced in [14] and its use for matching in Sect. 2. In Sect. 3 we show our algorithm for Swap Matching problem and follow it in Sect. 4 with the proof that Swap Matching cannot be solved efficiently by deterministic finite automata. Then we describe the Smalgo in detail in Sect. 5 and finish with the experimental evaluation of the algorithms in Sect. 6.

2 Basic Definitions and the Graph Theoretic Model

In this section we state the basic definitions, present the graph theoretic model and show a basic algorithm that solves Swap Matching using the model.

2.1 Notations and Basic Definitions

We use the word-RAM as our computational model. That means we have access to memory cells of fixed capacity w (e.g., 64 bits). A standard set of arithmetic and bitwise instructions include And (&), Or (|), Left bitwise-shift (LShift) and Right bitwise-shift (RShift). Each of the standard operations on words takes single unit of time. In order to compare to other existing algorithms, which are not streaming, we define the access to the input in a less restrictive way – the input is read from a read-only part of memory and the output is written to a write-only part of memory. However, it will be easy to observe that our algorithm accesses the input sequentially. We do not include the input and the output into the space complexity analysis.

A *string* S over an alphabet Σ is a finite sequence of symbols from Σ and $|S|$ is its length. By S_i we mean the i-th symbol of S and we define a *substring*

[1] http://users.fit.cvut.cz/blazeva1/gsm.html.

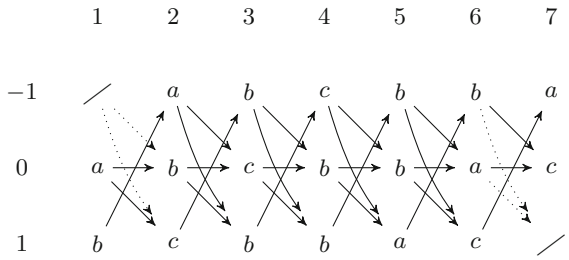


Fig. 1. P-graph $\mathcal{P}_P$ for the pattern $P = abcbbac$

$S_{[i,j]} = S_i S_{i+1} \dots S_j$ for $1 \leq i \leq j \leq |S|$, and *prefix* $S_{[1,i]}$ for $1 \leq i \leq |S|$. String P *prefix matches* string T n symbols on position i if $P_{[1,n]} = T_{[i,i+n-1]}$.

Next we formally introduce a swapped version of a string.

Definition 1 (Campanelli et al. [7]). *A* swap permutation *for S is a permutation $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$ such that:*

(i) if $\pi(i) = j$ then $\pi(j) = i$ (symbols at positions i and j are swapped),
(ii) for all $i, \pi(i) \in \{i-1, i, i+1\}$ (only adjacent symbols are swapped),
(iii) if $\pi(i) \neq i$ then $S_{\pi(i)} \neq S_i$ (identical symbols are not swapped).

For a string S a swapped version $\pi(S)$ *is a string $\pi(S) = S_{\pi(1)} S_{\pi(2)} \dots S_{\pi(n)}$ where π is a swap permutation for S.*

Now we formalize the version of matching we are interested in.

Definition 2. *Given a text $T = T_1 T_2 \dots T_t$ and a pattern $P = P_1 P_2 \dots P_p$, the pattern P is said to* swap match *T at location i if there exists a swapped version $\pi(P)$ of P that matches T at location i, i.e., $\pi(P) = T_{[i,i+p-1]}$.*

2.2 A Graph Theoretic Model

The algorithms in this paper are based on a model introduced by Iliopoulos and Rahman [14]. In this section we briefly describe this model.

For a pattern P of length p we construct a labeled graph $\mathcal{P}_P = (V, E, \sigma)$ with vertices V, edges E, and a vertex labeling function $\sigma : V \to \Sigma$ (see Fig. 1 for an example). Let $V = V' \setminus \{m_{-1,1}, m_{1,p}\}$ where $V' = \{m_{r,c} \mid r \in \{-1, 0, 1\}, c \in \{1, 2, \dots, p\}\}$. For $m_{r,c} \in V$ we set $\sigma(m_{r,c}) = P_{r+c}$. Each vertex $m_{r,c}$ is identified with an element of a $3 \times p$ grid. We set $E' := E'_1 \cup E'_2 \cup \dots \cup E'_{p-1}$, where $E'_j := \{(m_{k,j}, m_{i,j+1}) \mid k \in \{-1, 0\}, i \in \{0, 1\}\} \cup \{(m_{1,j}, m_{-1,j+1})\}$, and let $E = E' \cap V \times V$. We call $\mathcal{P}_P$ the *P-graph*. Note that $\mathcal{P}_P$ is directed acyclic graph, $|V(\mathcal{P}_P)| = 3p - 2$, and $|E(\mathcal{P}_P)| = 5(p-1) - 4$.

The idea behind the construction of $\mathcal{P}_P$ is as follows. We create vertices V' and edges E' which represent every swap pattern without unnecessary restrictions (equal symbols can be swapped). We remove vertices $m_{-1,1}$ and $m_{1,p}$ which represent symbols from invalid indices 0 and $p + 1$.

Algorithm 1. The basic matching algorithm (BMA)

Input: Labeled directed acyclic graph $G = (V, E, \sigma)$, set $Q_0 \subseteq V$ of starting vertices, set $F \subseteq V$ of accepting vertices, text T, and position k.

1: Let $D'_1 := Q_0$.
2: **for** $i = 1, 2, 3, \ldots, p$ **do**
3: Let $D_i := \{x \mid x \in D'_i, \sigma(x) = T_{k+i-1}\}$.
4: **if** $D_i = \emptyset$ **then** finish.
5: **if** $D_i \cap F \neq \emptyset$ **then** we have found a match and finish.
6: Define the next iteration set D'_{i+1} as vertices which are successors of D_i, i.e., $D'_{i+1} := \{d \in V(\mathcal{P}_P) \mid (v, d) \in E(\mathcal{P}_P)$ for some $v \in D_i\}$.

The P-graph now represents all possible swap permutations of the pattern P in the following sense. Vertices $m_{0,j}$ represent ends of prefixes of swapped version of the pattern which end by a non-swapped symbol. Possible swap of symbols P_j and P_{j+1} is represented by vertices $m_{1,j}$ and $m_{-1,j+1}$. Edges represent symbols which can be consecutive. Each path from column 1 to column p represents a swap pattern and each swap pattern is represented this way.

Definition 3. *For a given Σ-labeled directed acyclic graph $G = (V, E, \sigma)$ vertices $s, e \in V$ and a directed path $f = v_1, v_2, \ldots, v_k$ from $v_1 = s$ to $v_k = e$, we call $S = \sigma(f) = \sigma(v_1)\sigma(v_2)\ldots\sigma(v_k) \in \Sigma^*$ a* path string *of f.*

2.3 Using Graph Theoretic Model for Matching

In this section we describe an algorithm called *Basic Matching Algorithm* (BMA) which can determine whether there is a match of pattern P in text T on a position k using any graph model which satisfies the following conditions.

- It is a directed acyclic graph,
- $V = V_1 \uplus V_2 \uplus \cdots \uplus V_p$ (we can divide vertices to columns),
- $E = \{(u, w) \mid u \in V_i, w \in V_{i+1}, 1 \leq i < p\}$ (edges lead to next column).

Let $Q_0 = V_1$ be the *starting vertices* and $F = V_p$ be the *accepting vertices*. BMA is designed to run on any graph which satisfies these conditions. Since P-graph satisfies these assumptions we can use BMA for $\mathcal{P}_P$.

The algorithm runs as follows (see also Algorithm 1). We initialize the algorithm by setting $D'_1 := Q_0$ (Step 1). D'_1 now holds information about vertices which are the end of some path f starting in Q_0 for which $\sigma(f)$ possibly prefix matches 1 symbol of $T_{[k,k+p-1]}$. To make sure that the path f represents a prefix match we need to check whether the label of the last vertex of the path f matches the symbol T_k (Step 3). If no prefix match is left we did not find a match (Step 4). If some prefix match is left we need to check whether we already have a complete match (Step 5). If the algorithm did not stop it means that we have some prefix match but it is not a complete match yet. Therefore we can try to extend this prefix match by one symbol (Step 6) and check whether it is

Algorithm 2. BMA in terms of prefix match signals

1: Let $I^0(v) := 1$ for each $v \in Q_0$ and $I^0(v) := 0$ for each $v \notin Q_0$.
2: **for** $i = 0, 1, 2, 3, \ldots, p-1$ **do**
3: Filter signals by a symbol T_{k+i}.
4: **if** $I^i(v) = 0$ for every $v \in \mathcal{P}_P$ **then** finish.
5: **if** $I^i(v) = 1$ for any $v \in F$ **then** we have found a match and finish.
6: Propagate signals along the edges.

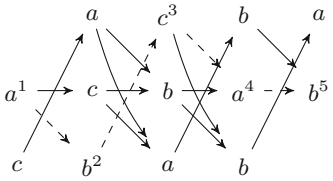


Fig. 2. BMA of $T_{[2,6]} = abcab$ on a P-graph of the pattern $P = acbab$. The prefix match signal propagates along the dashed edges. Index j above a vertex v represent that $I^j(v) = 1$, otherwise $I^j(v) = 0$.

a valid prefix match (Step 3). Since we extend the matched prefix in each step, we repeat these steps until the prefix match is as long as the pattern (Step 2).

Having vertices in sets is not handy for computing so we present another way to describe this algorithm. We use their characteristic vectors instead.

Definition 4. *A Boolean labeling function $I : V \to \{0, 1\}$ of vertices of $\mathcal{P}_P$ is called a* prefix match signal.

The algorithm can be easily divided into *iterations* according to the value of i in Step 2. We denote the value of the prefix match signal in j-th iteration as I^j and we define the following operations:

- *propagate signal along the edges*, is an operation which sets $I^j(v) := 1$ if and only if there exists an edge $(u, v) \in E$ with $I^{j-1}(u) = 1$,
- *filter signal by a symbol $x \in \Sigma$*, is an operation which sets $I(v) := 0$ for each v where $\sigma(v) \neq x$,
- *match check*, is an operation which checks whether there exists $v \in F$ such that $I(v) = 1$ and if so reports a match.

With these definitions in hand we can describe BMA in terms of prefix match signals as Algorithm 2. See Fig. 2 for an example of use of BMA to figure out whether $P = acbab$ swap matches $T = babcabc$ at a position 2.

2.4 Shift-And Algorithm

The following description is based on [9, Chap. 5] describing the Shift-Or algorithm.

For a pattern P and a text T of length p and t, respectively, let R be a bit array of size p and R^j its value after text symbol T_j has been processed. It contains information about all matches of prefixes of P that end at the position j in the text. For $1 \leq i \leq p$, $R_i^j = 1$ if $P_{[1,i]} = T_{[j-i+1,j]}$ and 0 otherwise. The vector R^{j+1} can be computed from R^j as follows. For each positive i we have $R_{i+1}^{j+1} = 1$ if $R_i^j = 1$ and $P_{i+1} = T_{j+1}$, and $R_{i+1}^{j+1} = 0$ otherwise. Furthermore, $R_1^{j+1} = 1$ if $P_1 = T_{j+1}$ and 0 otherwise. If $R_p^{j+1} = 1$ then a complete match can be reported.

The transition from R^j to R^{j+1} can be computed very fast as follows. For each $x \in \Sigma$ let D^x be a bit array of size p such that for $1 \leq i \leq p, D_i^x = 1$ if and only if $P_i = x$. The array D^x denotes the positions of the symbol x in the pattern P. Each D^x can be preprocessed before the search. The computation of R^{j+1} is then reduced to three bitwise operations, namely $R^{j+1} = (\text{LShift}(R^j) \mid 1)\ \&\ D^{T_{j+1}}$. When $R_p^j = 1$, the algorithm reports a match on a position $j - p + 1$.

3 Our Algorithm

In this section we will show an algorithm which solves Swap Matching. We call the algorithm GSM (Graph Swap Matching). GSM uses the graph theoretic model presented in Sect. 2.2 and is based on the Shift-And algorithm from Sect. 2.4.

The basic idea of the GSM algorithm is to represent prefix match signals (see Definition 4) from the basic matching algorithm (Sect. 2.3) over $\mathcal{P}_P$ in bit vectors. The GSM algorithm represents all signals I in the bitmaps RX formed by three vectors, one for each row. Each time GSM processes a symbol of T, it first propagates the signal along the edges, then filters the signal and finally checks for matches. All these operations can be done very quickly thanks to bitwise parallelism.

First, we make the concept of GSM more familiar by presenting a way to interpret the Shift-And algorithm by means of the basic matching algorithm (BMA) from Sect. 2.3 to solve the (ordinary) Pattern Matching problem. Then we expand this idea to Swap Matching by using the graph theoretic model.

3.1 Graph Theoretic View of the Shift-And Algorithm

Let T and P be a text and a pattern of lengths t and p, respectively. We create the T-graph $\mathcal{T}_P = (V, E, \sigma)$ of the pattern P.

Definition 5. *Let S be a string. The T-graph of S is a graph $\mathcal{T}_S = (V, E, \sigma)$ where $V = \{v_i \mid 1 \leq i \leq |S|\}$, $E = \{(v_i, v_{i+1}) \mid 1 \leq i \leq |S - 1|\}$ and $\sigma : V \to \Sigma$ such that $\sigma(v_i) = S_i$.*

We know that the T-graph is directed acyclic graph which can be divided into columns $V_i, 1 \leq i \leq p$ (each of them containing one vertex v_i) such that the edges lead from V_j to V_{j+1}. This means that the T-graph satisfies all assumptions of

BMA. We apply BMA to $\mathcal{T}_P$ to figure out whether P matches T at a position j. We get a correct result because for each $i \in \{1, \ldots, p\}$ we check whether $T_{j+i-1} = \sigma(v_i) = P_i$.

To find every occurrence of P in T we would have to run BMA for each position separately. This is basically the naive approach to solve the pattern matching. We can improve the algorithm significantly when we parallelize the computations of p runs of BMA in the following way.

The algorithm processes one symbol at a time starting from T_1. We say that the algorithm is in the j-th step when a symbol T_j has been processed. BMA represents a prefix match as a prefix match signal $I : V \to \{0, 1\}$. Its value in the j-th step is denoted I^j. Since one run of the BMA uses only one column of the T-graph at any time we can use other vertices to represent different runs of the BMA. We represent all prefix match indicators in one vector so that we can manipulate them easily. To do that we prepare a bit vector R. Its value in j-th step is denoted R^j and defined as $R^j_i = I^j(v_i)$.

First operation which is used in BMA (propagate signal along the edges) can be done easily by setting the signal of v_i to value of the signal of its predecessor v_{i-1} in the previous step. I.e., for $i \in \{1, \ldots, p\}$ we set $I^j(v_i) = 1$ if $i = 1$ and $I^j(v_i) = I^{j-1}(v_{i-1})$ otherwise. In terms of R^j this means just $R^j = \mathrm{LSO}(R^{j-1})$, where LSO is defined as $\mathrm{LSO}(x) = \mathrm{LShift}(x) \mid 1$.

We also need a way to set $I(v_i) := 0$ for each v_i for which $\sigma(v_i) \neq T_{j+i}$ which is another basic BMA operation (filter signal by a symbol). We can do this using the bit vector D^x from Sect. 2.4 and taking $R \,\&\, D^x$. I.e., the algorithm computes R^j as $R^j = \mathrm{LSO}(R^{j-1}) \,\&\, D^{T_{j+1}}$.

The last BMA operation we have to define is the *match detection*. We do this by checking whether $R^j_p = 1$ and if this is the case then a match starting at position $j - p + 1$ occurred.

3.2 Our Algorithm for Swap Matching Using the Graph Theoretic Model

Now we are ready to describe the GSM algorithm.

We again let $\mathcal{P}_P = (V, E, \sigma)$ be the P-graph of the pattern P, apply BMA to $\mathcal{P}_P$ to figure out whether P matches T at a position j, and parallelize p runs of BMA on $\mathcal{P}_P$.

Again, the algorithm processes one symbol at a time and it is in the j-th step when a symbol T_j is being processed. We again denote the value of the prefix match signal $I : V \to \{0, 1\}$ of BMA in the j-th step by I^j. I.e., the semantic meaning of $I^j(m_{r,c})$ is that $I^j(m_{r,c}) = 1$ if there exists a swap permutation π such that $\pi(c) = c + r$ and $\pi(P)_{[1,c]} = T_{[j-c+1,j]}$. Otherwise $I^j(m_{r,c})$ is 0.

We want to represent all prefix match indicators in vectors so that we can manipulate them easily. We can do this by mapping the values of I for rows $r \in \{-1, 0, 1\}$ of the P-graph to vectors RU, RM, and RD, respectively. We denote value of the vector $RX \in \{RU, RM, RD\}$ in j-th step as RX^j. We define values of the vectors as $RU^j_i = I^j(m_{-1,i})$, $RM^j_i = I^j(m_{0,i})$, and $RD^j_i = I^j(m_{1,i})$, where the value of $I^j(v) = 0$ for every $v \notin V$.

Algorithm 3. The graph swap matching (GSM)

Input: Pattern P of length p and text T of length t over alphabet Σ.
Output: Positions of all swap matches.
1: Let $RU^0 := RM^0 := RD^0 := 0^p$.
2: Let $D^x := 0^p$, for all $x \in \Sigma$.
3: **for** $i = 1, 2, 3, \ldots, p$ **do** $D^{P_i}_i := 1$
4: **for** $j = 1, 2, 3, \ldots, t$ **do**
5: $RU'^j := \mathrm{LSO}(RD^{j-1})$, $RM'^j := \mathrm{LSO}(RM^{j-1} \mid RU^{j-1})$
6: $RD'^j := \mathrm{LSO}(RM^{j-1} \mid RU^{j-1})$.
7: $RU^j := RU'^j \& \mathrm{LShift}(D^{T_j})$, $RM^j := RM'^j \& D^{T_j}$, $RD^j := RD'^j \& \mathrm{RShift}(D^{T_j})$.
8: **if** $RU^j_p = 1$ or $RM^j_p = 1$ **then** report a match on position $j - p + 1$.

We define *BMA propagate signal along the edges* operation as setting the signal of $m_{r,c}$ to 1 if at least one of its predecessors have signal set to 1. I.e., we set $I^{j+1}(m_{-1,i}) := I^j(m_{1,i-1})$, $I^{j+1}(m_{0,i}) := I^j(m_{-1,i-1}) \mid I^j(m_{0,i-1})$, $I^{j+1}(m_{0,1}) := 1$, $I^{j+1}(m_{1,i}) := I^j(m_{-1,i-1}) \mid I^j(m_{0,i-1})$, and $I^{j+1}(m_{1,1}) := 1$. We can perform the above operation using the $\mathrm{LSO}(R)$ operation. We obtain the *propagate signal along the edges* operation in the form $RU'^{j+1} := \mathrm{LSO}(RD^j)$, $RM'^{j+1} := \mathrm{LSO}(RM^j \mid RU^j)$, and $RD'^{j+1} := \mathrm{LSO}(RM^j \mid RU^j)$.

The operation *filter signal by a symbol* can be done by first constructing a bit vector D^x for each $x \in \Sigma$ as $D^x_i = 1$ if $x = P_i$ and $D^x_i = 0$ otherwise. Then we use these vectors to filter signal by a symbol x by taking $RU^j := RU'^j \& \mathrm{LShift}(D^{T_j})$, $RM^j := RM'^j \,\&\, D^{T_j}$, and $RD^j := RD'^j \,\&\, \mathrm{RShift}(D^{T_j})$.

The last operation we define is the match detection. We do this by checking whether $RU^j_p = 1$ or $RM^j_p = 1$ and if this is the case, then a match starting at a position $j - p + 1$ occurred.

The final GSM algorithm (Algorithm 3) first prepares the D-masks D^x for every $x \in \Sigma$ and initializes $RU^0 := RM^0 := RD^0 := 0$ (Steps 1–3). Then the algorithm computes the value of vectors RU^j, RM^j, and RD^j for $j \in \{1, \ldots, t\}$ by first using the above formula for signal propagation (Steps 5 and 6) and then the formula for signal filtering (Step 7) and checks whether $RU^j_p = 1$ or $RM^j_p = 1$ and if this is the case the algorithm reports a match (Step 8).

Observe that Algorithm 3 accesses the input sequentially and thus it is a streaming algorithm. We now prove correctness of our algorithm. To ease the notation let us define $R^j(m_{r,c})$ to be RU^j_c if $r = -1$, RM^j_c if $r = 0$, and RD^j_c if $r = 1$. We define $R'^j(m_{r,c})$ analogously. Similarly, we define $D^x(m_{r,c})$ as $(\mathrm{LShift}(D^x))_c = D^x_{c-1}$ if $r = -1$, D^x_c if $r = 0$, and $(\mathrm{RShift}(D^x))_c = D^x_{c+1}$ if $r = 1$. By the way the masks D^x are computed on lines 2 and 3 of Algorithm 3, we get the following observation.

Observation 1. *For every $m_{r,i} \in V$ and every $j \in \{i, \ldots t\}$ we have $D^{T_j}(m_{r,i}) = 1$ if and only if $T_j = P_{r+i}$.*

The following lemma constitutes the crucial part of the correctness proof.

Lemma 1. *For every $m_{r,i} \in V$ and every $j \in \{i, \dots t\}$ we have $R^j(m_{r,i}) = 1$ if and only if there exists a swap permutation π such that $\pi(P)_{[1,i]} = T_{[j-i+1,j]}$ and $\pi(i) = i + r$.*

Due to space constraints, we defer the proof of this lemma to the full version of this paper, see also its ArXiv version [6].

Our GSM algorithm reports a match on position $j - p + 1$ if and only if $R^j(m_{p,-1}) = 1$ or $R^j(m_{p,0}) = 1$. However, by Lemma 1, this happens if and only if there is a swap match of P on position $j - p + 1$ in T. Hence, the algorithm is correct.

Theorem 2. *The GSM algorithm runs in $O(\lceil \frac{p}{w} \rceil (|\Sigma| + t) + p)$ time and uses $O(\lceil \frac{p}{w} \rceil |\Sigma|)$ memory cells (not counting the input and output cells), where t is the length of the input text, p length of the input pattern, w is the word-size of the machine, and $|\Sigma|$ size of the alphabet.*[2]

Proof. The initialization of RX and D^x masks (lines 1 and 2) takes $O(\lceil \frac{p}{w} \rceil |\Sigma|)$ time. The bits in D^x masks are set according to the pattern in $O(p)$ time (line 3). The main cycle of the algorithm (lines 4–8) makes t iterations. Each iteration consists of computing values of RX in 13 bitwise operations, i.e., in $O(\lceil \frac{p}{w} \rceil)$ machine operations, and checking for the result in $O(1)$ time. This gives $O(\lceil \frac{p}{w} \rceil (|\Sigma|+t)+p)$ time in total. The algorithm saves 3 RX masks (using the same space for all j and also for RX' masks), $|\Sigma|$ D^x masks, and constant number of variables for other uses (iteration counters, temporary variable, etc.). Thus, in total the GSM algorithm needs $O(\lceil \frac{p}{w} \rceil |\Sigma|)$ memory cells. □

Corollary 1. *If $p = cw$ for some constant c, then the GSM algorithm runs in $O(|\Sigma| + p + t)$ time and has $O(|\Sigma|)$ space complexity. Moreover, if $p \leq w$, then the GSM algorithm can be implemented using only $7 + |\Sigma|$ memory cells.*

Proof. The first part follows directly from Theorem 2. Let us show the second part. We need $|\Sigma|$ cells for all D-masks, 3 cells for R vectors (reusing the space also for R' vectors), one pointer to the text, one iteration counter, one constant for the match check and one temporary variable for the computation of the more complex parts of the algorithm. Altogether, we need only $7 + |\Sigma|$ memory cells to run the GSM algorithm. □

From the space complexity analysis we see that for some sufficiently small alphabets (e.g. DNA sequences) the GSM algorithm can be implemented in practice using solely CPU registers with the exception of text which has to be loaded from the RAM.

[2] To simplify the analysis, we assume that $\log t < w$, i.e., the iteration counter fits into one memory cell.

4 Limitations of the Finite Deterministic Automata Approach

Many of the string matching problems can be solved by finite automata. The construction of a non-deterministic finite automaton that solves Swap Matching can be done by a simple modification of the P-graph. An alternative approach to solve the Swap Matching would thus be to determinize and execute this automaton. The drawback is that the determinization process may lead to an exponential number of states. We show that in some cases it actually does, contradicting the conjecture of Holub [13], stating that the number of states of this determinized automaton is $O(p)$.

Theorem 3. *There is an infinite family F of patterns such that any deterministic finite automaton A_P accepting the language $L_S(P) = \{u\pi(P) \mid u \in \Sigma^*, \pi$ is a swap permutation for $P\}$ for $P \in F$ has $2^{\Omega(|P|)}$ states.*

Proof. For any integer k we define the pattern $P_k := ac(abc)^k$. Note that the length of P_k is $\Theta(k)$. Suppose that the automaton A_P recognizing language $L(P)$ has s states such that $s < 2^k$. We define a set of strings $T_0, \ldots, T_{2^k-1}$ where T_i is defined as follows. Let $b^i_{k-1}, b^i_{k-2} \ldots b^i_0$ be the binary representation of the number i. Let $B^i_j = abc$ if $b^i_j = 0$ and let $B^i_j = bac$ if $b^i_j = 1$. Then, let $T_i := acB^i_{k-1}B^i_{k-2}\ldots B^i_0$. See Table 1 for an example. Since $s < 2^k$, there exist $0 \leq i < j \leq 2^k - 1$ such that both T_i and T_j are accepted by the same accepting state q of the automaton A. Let m be the minimum number such that $b^i_{k-1-m} \neq b^j_{k-1-m}$. Note that $b^i_m = 0$ and $b^j_m = 1$. Now we define $T'_i = T_i(abc)^{(m+1)}$ and $T'_j = T_j(abc)^{(m+1)}$. Let $X = (T'_i)_{[3(m+1)+1,3(m+1+k)+2]}$ and $Y = (T'_j)_{[3(m+1)+1,3(m+1+k)+2]}$ be the suffices of the strings T'_i and T'_j both of length $3k+2$. Note that X begins with $bc\ldots$ and Y begins with $ac\ldots$ and that block abc or bac repeats for k times in both. Therefore pattern P swap matches Y and does not swap match X. Since for the last symbol of both T_i and T_j the automaton is in the same state q, the computation for T'_i and T'_j must end in the same state q'. However as X should not be accepted and Y should be accepted we obtain contradiction with the correctness of the automaton A. Hence, we may define the family F as $F = \{P_1, P_2, \ldots\}$, concluding the proof. □

Table 1. An example of the construction from proof of Theorem 3 for $k = 3$.

$P = T_0$	$acabcabcabc$
T_1	$acabcabcbac$
T_2	$acabcbacabc$
T_3	$acabcbacbac$
T_4	$acbacabcabc$
T_5	$acbacabcbac$
T_6	$acbacbacabc$
T_7	$acbacbacbac$

This proof shows the necessity for specially designed algorithms which solve the Swap Matching. We presented one in the previous section and now we reiterate on the existing algorithms.

5 Smalgo Algorithm

In this section we discuss how Smalgo by Iliopoulos and Rahman [14] and Smalgo-I by Ahmed et al. [1] work. Since they are bitwise inverse of each other, we will introduce them both in terms of operations used in Smalgo-I.

Before we show how these algorithms work, we need one more definition.

Definition 6. *A* degenerate symbol w *over an alphabet* Σ *is a nonempty set of symbols from alphabet* Σ*. A* degenerate string S *is a string built over an alphabet of degenerate symbols. We say that a degenerate string* $\widetilde{P}$ *matches a text* T *at a position* j *if* $T_{j+i-1} \in \widetilde{P}_i$ *for every* $1 \leq i \leq p$*.*

5.1 Smalgo-I

The Smalgo-I [1] is a modification of the Shift-And algorithm from Sect. 2.4 for Swap Matching. The algorithm uses the graph theoretic model introduced in Sect. 2.2.

First let $\widetilde{P} = \{P_1, P_2\} \dots \{P_{x-1}, P_x, P_{x+1}\} \dots \{P_{p-1}, P_p\}$ be a degenerate version of pattern P. The symbol in $\widetilde{P}$ on position i represents the set of symbols of P which can swap to that position. To accommodate the Shift-And algorithm to match degenerate patterns we need to change the way the D^x masks are defined. For each $x \in \Sigma$ let $\widetilde{D}_i^x$ be the bit array of size p such that for $1 \leq i \leq p$, $\widetilde{D}^x = 1$ if and only if $x \in \widetilde{P}_i$.

While a match of the degenerate pattern $\widetilde{P}$ is a necessary condition for a swap match of P, it is clearly not sufficient. The way the Smalgo algorithms try to fix this is by introducing *P-mask* $P(x_1, x_2, x_3)$ which is defined as $P(x_1, x_2, x_3)_i = 1$ if $i = 1$ or if there exist vertices u_1, u_2, and u_3 and edges $(u_1, u_2), (u_2, u_3)$ in $\mathcal{P}_P$ for which $u_2 = m_{r,i}$ for some $r \in \{-1, 0, 1\}$ and $\sigma(u_n) = x_n$ for $1 \leq n \leq 3$, and $P(x_1, x_2, x_3)_i = 0$ otherwise. One P-mask called $P(x, x, x)$ is used to represent the P-masks for triples (x_1, x_2, x_3) which only contain 1 in the first column.

Now, whenever checking whether P prefix swap matches T $k + 1$ symbols at position j we check for a match of $\widetilde{P}$ in T and we also check whether $P(T_{j+k-1}, T_{j+k}, T_{j+k+1})_{k+1} = 1$. This ensures that the symbols are able to swap to respective positions and that those three symbols of the text T are present in some $\pi(P)$.

With the P-masks completed we initialize $R^1 = 1 \& \widetilde{D}^{T_1}$. Then for every $j = 1$ to t we repeat the following. We compute R^{j+1} as $R^{j+1} = \text{LSO}(R^j)$ & $\widetilde{D}^{T_{j+1}}$ & $\text{RShift}(\widetilde{D}^{T_{j+2}})$ & $P(T_j, T_{j+1}, T_{j+2})$. To check whether or not a swap match occurred we check whether $R^j_{p-1} = 1$. This is claimed to be sufficient because during the processing we are in fact considering not only the next symbol T_{j+1} but also the symbol T_{j+2}.

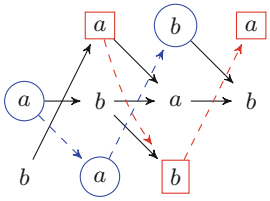


Fig. 3. SMALGO flaw represented in the P-graph for $P = abab$

5.2 The Flaw in the Smalgo, Smalgo-I and Smalgo-II

We shall see that for a pattern $P = abab$ and a text $T = aaba$ all SMALGO versions give false positives.

The concept of SMALGO is based on the assumption that we can find a path in $\mathcal{P}_P$ by searching for consecutive paths of length 3 (*triplets*), where each two consecutive share two columns and can partially overlap. However, this only works if the consecutive triplets *actually* share the two vertices in the common columns. If the assumption is not true then the found substring of the text might not match any swap version of P.

The above input gives such a configuration and therefore the assumption is false. The SMALGO-I algorithm actually reports match of pattern $P = abab$ on a position 1 of text $T = aaba$. This is obviously a false positive, as the pattern has two b symbols while the text has only one.

The reason behind the false positive match is as follows. The algorithm checks whether the first triplet of symbols (a, a, b) matches. It can match the swap pattern $aabb$. Next it checks the second triplet of symbols (a, b, a), which can match $baba$. We know that $baba$ is not possible since it did not appear in the previous check, but the algorithm cannot distinguish them since it only checks for triplets existence. Since each step gave us a positive match the algorithm reports a swap match of the pattern in the text.

In the Fig. 3 we see the two triplets which SMALGO assumes have two vertices in common. The SMALGO-II algorithm saves space by maintaining less information, however it simulates how SMALGO-I works and so it contains the same flaw. The ArXiv version of our paper [6] provides more details on the execution of SMALGO-I algorithm on pattern $P = abab$ and text $T = aaba$ and also a detailed analysis of the SMALGO-II algorithm.

6 Experiments

We implemented our Algorithm 3 (GSM), described in Sect. 3.2, the Bitwise Parallel Cross Sampling (BPCS) algorithm by Cantone and Faro [8], the Bitwise Parallel Backward Cross Sampling (BPBCS) algorithm by Campanelli et al. [7], and the faulty SMALGO algorithm by Iliopoulos and Rahman [14]. All these implementations are available online. (see footnote 1)

Table 2. Comparison of the running times. Each value is the average over 10,000 patterns randomly selected from the text in milliseconds.

Data ($\|\Sigma\|$)	Algor.	Pattern length									
		3	4	5	6	8	9	10	12	16	32
CH (5)	SMALGO	426	376	355	350	347	347	344	347	345	345
	BPCS	398	**353**	**335**	**332**	**329**	329	326	328	329	327
	BPBCS	824	675	555	472	366	**328**	**297**	**257**	**199**	**112**
	GSM	**394**	354	338	333	332	331	329	333	331	333
HS (19)	SMALGO	4.80	4.73	4.72	4.74	4.70	4.71	4.71	4.71	4.72	4.70
	BPCS	4.43	**4.36**	**4.36**	4.36	4.34	4.33	4.34	4.34	4.35	4.34
	BPBCS	7.16	5.80	4.79	**4.05**	**3.03**	**2.70**	**2.44**	**2.06**	**1.62**	**0.95**
	GSM	**4.42**	4.38	4.41	4.46	4.45	4.45	4.45	4.44	4.53	4.48
BIB (62)	SMALGO	8.60	8.38	8.29	8.34	8.32	8.33	8.30	8.35	8.35	8.33
	BPCS	7.53	**7.36**	**7.28**	7.29	7.26	7.27	7.26	7.28	7.29	7.25
	BPBCS	12.43	10.03	8.26	**7.03**	**5.44**	**4.93**	**4.52**	**3.93**	**3.19**	**1.88**
	GSM	**7.52**	7.37	7.31	7.35	7.38	7.40	7.38	7.42	7.44	7.40

We tested the implementations on three real-world datasets. The first dataset (CH) is the 7$^{\text{th}}$ chromosome of the human genome[3] which consists of 159 M characters from the standard `ACTG` nucleobases and `N` as for non-determined. Second dataset (HS) is a partial genome of Homo sapiens from the Protein Corpus[4] with 3.3 M characters representing proteins encoded in 19 different symbols. The last dataset (BIB) is the Bible text of the Cantenbury Corpus[5] with 4.0 M characters containing 62 different symbols. For each length from $3, 4, 5, 6, 8, 9, 10, 12, 16$, and 32 we randomly selected 10,000 patterns from each text and processed each of them with each implemented algorithm.

All measured algorithms were implemented in `C++` and compiled with `-O3` in `gcc 6.3.0`. Measurements were carried on an Intel Core i7-4700HQ processor with 2.4 GHz base frequency and 3.4 GHz turbo with 8 GiB of DDR3 memory at 1.6 GHz. Time was measured using `std::chrono::high_resolution_clock::now()` from the `C++ chrono` library. The resulting running times, shown in Table 2, were averaged over the 10,000 patterns of the given length.

The results show, that the GSM algorithm runs approximately 23% faster than Smalgo (ignoring the fact that Smalgo is faulty by design). Also, the performance of GSM and BPCS is almost indistinguishable and according to our experiments, it varies in the span of units of percents depending on the exact CPU, cache, RAM and compiler setting. The seemingly superior average performance of BPBCS is caused by the heuristics BPBCS uses; however, while the worst-case performance of GSM is guaranteed, the performance of BPBCS for

[3] ftp://ftp.ensembl.org/pub/release-90/fasta/homo_sapiens/dna/.
[4] http://www.data-compression.info/Corpora/ProteinCorpus/.
[5] http://corpus.canterbury.ac.nz/descriptions/large/bible.html.

Table 3. Found occurrences across datasets: The value is simply the sum of occurrences over all the patterns.

Algorithm	Dataset		
	CH	HS	BIB
SMALGO	86243500784	51136419	315612770
Rest	84411799892	51034766	315606151

certain patterns is worse than that of GSM. Also note that GSM is a streaming algorithm while the others are not.

Table 3 visualizes the accurateness of SMALGO-I with respect to its flaw by comparing the number of occurrences found by the respective algorithms. The ratio of false positives to true positives for the SMALGO-I was: CH 2.17%, HS 0.20% and BIB 0.002%.

References

1. Ahmed, P., Iliopoulos, C.S., Islam, A.S., Rahman, M.S.: The swap matching problem revisited. Theoret. Comput. Sci. **557**, 34–49 (2014)
2. Amir, A., Aumann, Y., Landau, G.M., Lewenstein, M., Lewenstein, N.: Pattern matching with swaps. J. Algorithms **37**(2), 247–266 (2000)
3. Amir, A., Cole, R., Hariharan, R., Lewenstein, M., Porat, E.: Overlap matching. Inf. Comput. **181**(1), 57–74 (2003)
4. Amir, A., Landau, G.M., Lewenstein, M., Lewensteint, N.: Efficient special cases of pattern matching with swaps. Inform. Process. Lett. **68**(3), 125–132 (1998)
5. Antoniou, P., Iliopoulos, C.S., Jayasekera, I., Rahman, M.S.: Implementation of a swap matching algorithm using a graph theoretic model. In: Elloumi, M., Küng, J., Linial, M., Murphy, R.F., Schneider, K., Toma, C. (eds.) BIRD 2008. CCIS, vol. 13, pp. 446–455. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70600-7_34
6. Blažej, V., Suchý, O., Valla, T.: A simple streaming bit-parallel algorithm for swap pattern matching. CoRR abs/1606.04763 (2017). http://arxiv.org/abs/1606.04763
7. Campanelli, M., Cantone, D., Faro, S.: A new algorithm for efficient pattern matching with swaps. In: Fiala, J., Kratochvíl, J., Miller, M. (eds.) IWOCA 2009. LNCS, vol. 5874, pp. 230–241. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10217-2_24
8. Cantone, D., Faro, S.: Pattern matching with swaps for short patterns in linear time. In: Nielsen, M., Kučera, A., Miltersen, P.B., Palamidessi, C., Tůma, P., Valencia, F. (eds.) SOFSEM 2009. LNCS, vol. 5404, pp. 255–266. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-95891-8_25
9. Charras, C., Lecroq, T.: Handbook of Exact String Matching Algorithms. King's College Publications, London (2004)
10. Chedid, F.: On pattern matching with swaps. In: AICCSA 2013, pp. 1–5. IEEE (2013)
11. Faro, S.: Swap matching in strings by simulating reactive automata. In: Proceedings of the Prague Stringology Conference 2013, pp. 7–20. CTU in Prague (2013)

12. Fredriksson, K., Giaquinta, E.: On a compact encoding of the swap automaton. Inf. Proces. Lett. **114**(7), 392–396 (2014)
13. Holub, J.: Personal communication (2015)
14. Iliopoulos, C.S., Rahman, M.S.: A new model to solve the swap matching problem and efficient algorithms for short patterns. In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM 2008. LNCS, vol. 4910, pp. 316–327. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77566-9_27
15. Lewin, B.: Genes for SMA: multum in parvo. Cell **80**(1), 1–5 (1995)
16. Muthukrishnan, S.: New results and open problems related to non-standard stringology. In: Galil, Z., Ukkonen, E. (eds.) CPM 1995. LNCS, vol. 937, pp. 298–317. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60044-2_50
17. Wagner, R.A., Lowrance, R.: An extension of the string-to-string correction problem. J. ACM **22**(2), 177–183 (1975)

Epidemic Intelligence Statistical Modelling for Biosurveillance

Christina Parpoula[1(✉)], Alex Karagrigoriou[1], and Angeliki Lambrou[2]

[1] Lab of Statistics and Data Analysis, Department of Mathematics, University of the Aegean, 83200 Karlovasi, Samos, Greece
parpoula.ch@aegean.gr

[2] Hellenic Center for Disease Control and Prevention, 15123 Athens, Greece

Abstract. The last two decades, the emergence of new infectious diseases and the occasional rapid increase of their cases worldwide, the intense concern about bioterrorism, pandemic influenza or/and other Public Health threats, and the increasing volumes of epidemiological data, are all key factors that made necessary the development of advanced biosurveillance systems. Additionally, these factors have resulted in the awakening of the scientific community for introducing new and more efficient epidemic outbreak detection methods. As seen from above, the biosurveillance is a dynamic scientific activity which progresses and requires systematic monitoring of developments in the field of health sciences and biostatistics. This paper deals with the development of statistical regression modelling techniques in order to provide guidelines for the selection of the optimal periodic regression model for early and accurate outbreak detection in an epidemiological surveillance system, as well as for its proper use and implementation.

Keywords: Statistical modelling · Data analysis
Periodic regression model · Computer science · Biosurveillance
Epidemiology

1 Introduction

1.1 Objectives/Goals

The health of the population is a valuable commodity in the center of interest of society and health services [23]. Continuous rapid changes in the environment and the socio-economic conditions, as well as the observed changes in the epidemiology of diseases and the burden they cause on humanity are the main axes that impose the necessity of infectious disease surveillance (see [8,15,25]). The major challenges in this expanding field derive from its focus on detection of outbreaks as they arise, in a sufficiently timely fashion to enable effective control measures to be taken. The process governing these issues is now referred to as biosurveillance [20].

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 349–363, 2017.
https://doi.org/10.1007/978-3-319-72453-9_29

A specific sector of biosurveillance related solely to the human population, is epidemiological surveillance, which can be defined as "the continuous, systematic collection, analysis and interpretation of health-related data needed for the planning, implementation, and evaluation of public health practice" [31]. Epidemiological surveillance aims at monitoring the distribution and trends of the incidence of diseases as well as their further processing in order to design, implement and evaluate health policies and public health actions [11], with the ultimate aim of reducing morbidity and mortality, and hence improve health indicators of the population (see [12,14]). The modern definition of epidemiological surveillance includes aspects such as controlling the validity of data, analyzing data via advanced statistical methods as well as extracting safe conclusions with scientific and methodological adequacy [7].

As seen from above, the biosurveillance is a dynamic scientific activity which progresses and requires systematic monitoring of developments in the field of health sciences and biostatistics. The need for timely and accurate prediction of the time occurrence of an epidemic wave remains, and has led to further developments, as also evidenced by the implementation of statistical routine methods to detect outbreaks in biosurveillance systems in several European countries [10] and Centers for Disease Control and Prevention (CDC). Biosurveillance systems present various challenges regarding the source of data, the statistical quality control, the monitoring (follow-up), the evaluation of statistical techniques used to detect outbreaks, anomalies and outliers in biosurveillance data, and extreme timeliness of detection.

The combination of a new requirement of timeliness, a high level of applied work building early warning systems, and a set of research questions that need to be addressed to facilitate the applied work suggests that it may be beneficial to view valid and early detection of outbreaks as an area of new scientific inquiry and to ask what the theoretical underpinnings of this field are, what constitutes the relevant existing body of knowledge, and how this scientific field can facilitate the applied work. Towards this end, this paper aims at the development of cutting edge methodology for the automated, efficient, accurate, very early detection and modelling of outbreaks in biosurveillance systems. More specifically, the goals of the paper are described as follows: 1. to review in brief the literature and identify the mathematical foundations of early outbreak detection; 2. to develop regression approaches (within this framework, periodic regression models that work best for monitoring processes will be recommended); 3. to evaluate (qualitatively and quantitatively) and compare the developed methodologies by conducting a retrospective analysis of epidemiologic time series.

1.2 State of the Art

Perhaps the simplest regression model for outbreak detection is that described by Stroup et al. in [24], which ensures that seasonal effects are automatically adjusted for by design rather than by explicit modelling, thus providing some element of robustness. However, this model does not incorporate time trends. A commonly used fully parametric outbreak detection regression model is based on

that of Serfling in [19], who modelled historical baselines using a trigonometric function with linear trend, assuming normal errors with constant variance. This model has subsequently been used to detect the onset of epidemics of influenza by Costagliola et al. in [5] and in [4]. An automated version of the Serfling model with cubic trend and three trigonometric terms, with model selection based on the Akaike Information Criterion (AIC), has been developed for prospective and retrospective surveillance by Pelat et al. in [17].

As documented above, a rich array of statistical methodologies is available for the early detection of epidemic activity in biosurveillance systems. This alone raises the research question "Which is the most appropriate methodology to use?" It is not feasible to make detailed recommendations as to which method is "best", because this depends critically on the specific details of the application and implementation, as well as its purpose and context. During the past decade, a rapid growth of research interest was observed as well as a widespread use of advanced statistical modelling techniques (regression models, time series models, Bayesian and Markov models, spatio-temporal models) for biosurveillance purposes. In the relevant areas of biostatistics and biosurveillance, we find several detailed reviews of these statistical methods in the recent literature, such as those of Sonesson and Bock in [22], Farrington and Andrews in [6], Buckeridge et al. in [2], Shmueli and Burkom in [20], and Unkel et al. in [27]. This paper focuses on the study of parametric statistical periodic regression models aiming at the early and accurate detection of outbreaks in epidemiological surveillance systems.

2 Materials and Methods

2.1 Sentinel Epidemiological Surveillance System

In Greece, since 1999, a system of epidemiological surveillance (sentinel surveillance system) is working and is based on voluntary participation of physicians, general practitioners and pediatricians of Primary Health Care (PHC) throughout Greece. The sentinel systems in PHC through registration, processing, analysis and results/conclusions export procedures, provide general guidelines for optimal decision making in health services. Sentinel systems are the most important source of primary care epidemiological diseases data. Through the sentinel system, the evolution of the frequency of certain diseases is recorded by carefully selected reporting sites and health workers who report cases of the disease or syndrome under surveillance, based on clinical diagnoses. In particular, the sentinel medical doctors send weekly epidemiological data regarding the number of consultations for all causes and the number of consultations for each syndrome under surveillance, according to a specified clinical definition. These reporting forms enable the Hellenic Center for Disease Control and Prevention (HCDCP) to estimate the weekly number of syndrome cases per 1,000 visits, i.e., the proportional morbidity, which reflects the activity of the syndrome under study.

After the completion of the reorganization and the changes in the operating parameters of the sentinel system, during the period 2014–2015, under the Operational Programme "Human Resources Development" of the National Strategic Reference Framework (NSRF) 2007–2013, action "Primary Health Care services (private and public) networking for epidemiological surveillance and control of communicable diseases", the national priorities were posed again for the syndromes monitored through the sentinel system. The influenza-like illness (ILI) and gastroenteritis were established as syndromes of main interest. The study of the evolution of these two syndromes is a major public health concern, since despite the fact that they belong to the sentinel epidemiological surveillance priorities of the country, are also monitored traditionally by sentinel systems in the European region, while they are high in terms of international interest, due to their potential for widespread transmission (with ILI also representing a potential pandemic risk). Moreover, the surveillance of these two syndromes through the sentinel system allows studying the existence of seasonality, the determination of the signaled start and end weeks and the intensity of epidemic waves for influenza-like illness, as well as the determination of epidemic outbreaks for gastroenteritis nationwide.

2.2 Two Season Influenza Historical Data

As aforementioned, through the sentinel system, the evolution of the frequency of certain diseases is recorded by carefully sampled reporting units, based on clinical diagnoses. These include influenza, or better, clinical manifestations compatible with flu, i.e., ILI. In the context of the reorganization of the sentinel system in Greece, the system was harmonized with the updated European and international standards-instructions. At the same time though, the recent sentinel surveillance system data (week40/2014 to date) are not considered comparable to those of previous years (past influenza seasons until week39/2014). Thus, this paper focuses on the study of weekly ILI rate data between September 29, 2014 and October 2, 2016, which were used for analysis purposes, in order to determine the signaled start and end weeks for the past two seasonal influenza outbreaks, and also to establish optimal empirical epidemic thresholds. Therefore, we conducted a retrospective analysis for the period from 2014 to 2016, based on a model fit to two season historical data (week40/2014 to week39/2016) with the main objectives being the prediction of the time interval for which an influenza outbreak is expected, and the estimation of its duration based on the available past data, as well as the early detection of possible epidemics.

2.3 Research Methodology

Two types of analysis exist for surveillance time series: retrospective analysis, to locate and quantify the impact of past epidemics, and prospective analysis, for real time detection of epidemics [17]. This paper focuses on retrospective analysis, epidemic detection and quantification from time series data. In such a

case (detection of influenza epidemics in time series), four steps are necessary to be followed:

1. Determination of the training period: a subset of data, i.e., training data, is selected from the whole time series to estimate the baseline level;
2. Purge of the training period: a rule is used to selectively discard epidemic events from the training data, so that the baseline level is estimated from truly non epidemic data;
3. Estimation of the regression equation: a periodic regression model is fitted to the training data;
4. Epidemic alert notification: the model is used to define an epidemic threshold and/or estimate excess morbidity.

The relevant R codes that have been appropriately adjusted are freely available in "Additional file 1" by Pelat et al. in [17].

Determination of the Training Period. It is not generally the case that all data should be included in the training period even if long time series are available [18]. Actually, as discussed by Pelat et al. in [17], changes in case reporting or/and demographics will likely be present over long time periods, and this fact may affect how well the baseline model fits the data. Modelling of influenza morbidity or/and mortality typically uses the five preceding years in baseline determination. In this paper, all data were included in the training period, and thus we used the whole dataset in the model fitting for retrospective analysis (as done, for example, in [17,21,30]), since after the completion of the reorganization of the sentinel system in Greece, the recent two season historical data (week40/2014 to week39/2016) are not considered comparable to those of previous years. Including more past seasons improves the seasonal components estimates, while limiting the quantity of data allows capturing recent trends [17]. A minimum of one year historical data is required to fit the models, but more reliable predictions require at least two or three year historical data to calculate the baseline level, as pointed out by Pelat et al. in [17].

Purge of the Training Period. The model must be fitted on non-epidemic data in order to model the non-epidemic baseline level. For seasonal diseases such as influenza, epidemics typically occur every year, thus it is difficult to find long epidemic-free periods. There exist two alternative choices to deal with the presence of epidemics in the training data, as discussed in [17]. The first choice is to identify epidemics, and then exclude the corresponding data from the series, whereas, the second and less common choice, requires explicit modelling of the epidemic periods during the training data. In the latter case, an epidemic indicator must be included as a covariable in the model. However, the availability of an independent epidemic indicator is uncommon in practice as pointed out by Pelat et al. in [17].

In the first choice, epidemics must first be identified, and several rules have been suggested in the literature in this respect, such as excluding the 25% highest

values from the training period [30], removing all data above a given threshold (more than three influenza-like illness cases per sentinel general practitioner) [5], or excluding the months with "reported increased respiratory disease activity or a major mortality event" [16]. In other studies, entire epidemic periods are deleted, for example December to April [21], or September to mid-April [28]. In this paper, we selected to exclude the 15% highest observations from the training period, that is the default value selected by Pelat et al. in [17].

Estimation of the Regression Equation. Several different formulations may be used for the regression equation, such as linear regression [5], linear regression on the log-transformed series [1], Poisson regression [26], and Poisson regression allowing for over-dispersion [29]. In this paper, linear regression is applied on the available two season historical data series (weekly ILI rate data). The weekly estimated number of influenza-like syndrome cases per 1,000 visits (ILI rate) is a time series with specific characteristic properties, i.e., tendency (tends to increase/decrease for a certain time) and seasonality (seasonal variations). In the regression equation, the trend is usually modelled using a linear term or a polynomial (of 2nd or 3rd degree) [17]. Regarding the seasonality, it is generally modelled using sine and cosine terms with period one year, however, refined models are found in the literature, often with terms of period six months [5], sometimes 3 months [9], and, rarely, smaller [13].

In this paper, we follow an exhaustive search process in order to identify the optimal fit of the baseline model. Thus, linear, quadratic and cubic trends are considered, and regarding the seasonal component, the most widely used periodicities are implemented, i.e. 12, 6 and 3 months. Higher degree polynomials or alternative seasonal terms have not been considered due to the fact that they may be more prone to result in unidentifiable models or other problems with model fit [17]. As a result, the time period under study is the explanatory variable, the observed time series values (weekly ILI rate) is the dependent variable, and all regression equations for the observed value $Y(t)$ are special cases of the following model:

$$Y(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \alpha_3 t^3 + \gamma_1 \cos(\frac{2\pi t}{n}) + \delta_1 \sin(\frac{2\pi t}{n})$$
$$+\gamma_2 \cos(\frac{4\pi t}{n}) + \delta_2 \sin(\frac{4\pi t}{n}) + \gamma_3 \cos(\frac{8\pi t}{n}) + \delta_3 \sin(\frac{8\pi t}{n}) + \varepsilon(t), \qquad (1)$$

where $\varepsilon(t)$ are centered zero-mean random variables with variance σ^2, n denotes the sample size, and model coefficients are estimated by least squares regression.

Selection of the best fitting model is made possible by an exhaustive search and selection process which is relied on Analysis Of Variance (ANOVA) comparison (significance level is chosen to be 0.05) to select between nested models, and on Akaike's Criterion (AIC) or on Schwarz's Bayesian Information Criterion (BIC) criterion, to select between non-nested models. The latter process is described step-by-step as follows: The exhaustive process starts comparing, by ANOVA, the simplest model labelled as M11 ($Y(t) = \alpha_0 + \alpha_1 t + \gamma_1 \cos(\frac{2\pi t}{n}) +$

$\delta_1 \sin(\frac{2\pi t}{n}) + \varepsilon(t)$) with the two models in which it is nested, labelled as M12 ($Y(t) = \alpha_0 + \alpha_1 t + \gamma_1 \cos(\frac{2\pi t}{n}) + \delta_1 \sin(\frac{2\pi t}{n}) + \gamma_2 \cos(\frac{4\pi t}{n}) + \delta_2 \sin(\frac{4\pi t}{n}) + \varepsilon(t)$) and M21 ($Y(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \gamma_1 \cos(\frac{2\pi t}{n}) + \delta_1 \sin(\frac{2\pi t}{n}) + \varepsilon(t)$). In the case that none of the alternative models (M12 and M21) is significantly better than the initial one (M11), the process retains M11 and stops. If one of the two alternative models is better than the initial one, the algorithmic process keeps it and goes on. If both alternative models are better than the initial one, the algorithmic process keeps the one with the lowest AIC or BIC and goes on. The procedure is repeated until finding the "best overall" model over the nine considered models (combining the three choices for the trend and periodicity).

Epidemic Alert Notification. The standard deviation of the residuals (difference between observed and model value) may be used to estimate the variation around the model fit, as the baseline model is fitted to the observations. In this way, assuming that the baseline model holds in the future, it is possible to obtain forecast intervals for future observations [17]. The epidemic thresholds which signal an unexpected change are typically obtained by taking an upper percentile for the prediction distribution (assumed to be normal), typically the upper 95th percentile [5], or upper 90th percentile [21]. Increasing this value will lead to less observations outside the thresholds and more specific detection, whereas decreasing the threshold will increase sensitivity and timeliness of the alerts. In this paper, the epidemic threshold are obtained by taking the upper 95th percentile of the prediction distribution. A rule is then used to define when epidemic alerts are produced, such as "*a series of observations fall above the epidemic threshold*". The latter step is necessary to avoid making alerts for isolated data points, and a minimum duration above the epidemic threshold may therefore be required. In this paper, the rule was set to be "a series of observations fall above the epidemic threshold during 2 weeks" (see [17,30]). The beginning of the epidemic is the first time the series exceeds the threshold, and the end the first time the series returns below the threshold.

3 Experimental Study

We conducted a retrospective analysis; the whole time series, i.e., week40/2014 to week39/2016, was therefore included in the training period. Then, we chose to exclude the top 15% observations from the training period (89 kept values from the total of 105, 84.76% of the entire series). Models selected through the model selection pathway based on ANOVA comparison, and AIC or BIC criterion were M11 (with linear trend and annual periodicity), M12 (with linear trend, annual and semi-annual periodicity), M22 (with quadratic trend, annual and semi-annual periodicity) and M23 (with quadratic trend, annual, semi-annual and quarterly periodicity), using either AIC or BIC criterion. Using the algorithm pathway, the model final kept and chosen for baseline influenza morbidity, was **M23** with quadratic trend, and annual, semi-annual and quarterly periodic terms (one year and six months and 3 months harmonics), using either AIC or

BIC criterion. Then, the forecast interval was set to be 95%, that is the upper limit of the prediction interval which is used as a threshold to detect epidemics. Finally, the alert rule was chosen to be "an epidemic is declared when 2 weekly successive observations are above the estimated threshold".

The mathematical form of model **M23** is described as follows:

$$Y(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \gamma_1 \cos(\frac{2\pi t}{n}) + \delta_1 \sin(\frac{2\pi t}{n}) + \gamma_2 \cos(\frac{4\pi t}{n}) + \delta_2 \sin(\frac{4\pi t}{n}) + \gamma_3 \cos(\frac{8\pi t}{n}) + \delta_3 \sin(\frac{8\pi t}{n}) + \varepsilon(t). \quad (2)$$

The estimated parameters together with the standard errors (sd), the statistic values (t-value) and the associated p-values of the selected model are presented in Table 1. The nine periodic regression models are described in Table 2, in which the components included in each model are indicated by "*". Models selected through the algorithm pathway are in italics (M11, M12, M22 and M23), and the model finally kept (M23) is in bold italics. Table 2 also presents the AIC and BIC criteria values, as well as the multiple R^2 and adjusted R^2 values for each of the examined models. Figures 1 and 2 illustrate the model selection pathway, using AIC or BIC criterion, respectively. Models selected through the algorithm pathway are presented in italics and the model finally chosen appears in bold italics in Figs. 1 and 2.

Additionally, a comparative study was conducted in order to examine how the selected model (M23) compares to typical trend or/and seasonality detection models (discussed in Subsect. 1.2) in terms of prediction performance. The standard modelling techniques of a linear trend (LT), a simple moving average of 3 terms (MA3), a simple exponential smoothing with parameter equal to 0.1065 (SES), the Holt's linear exponential smoothing with parameters equal to 0.0981 & 0.0246 (Holt's model), the Brown's quadratic exponential smoothing with parameter equal to 0.0296 (Brown's model), the Winter's exponential smoothing with parameters equal to 0.1062, 0.1032 & 0.1036 (Winter's model), and the standard CDC algorithm for flu detection assuming the signal follows a sinusoid with a period of one year (Serfling's cyclic regression model), were examined. Evaluation criteria such as the Root Mean Squared Error (RMSE), AIC and BIC were employed in the comparative study. The choice of RMSE among other widely used metrics was based on the fact that, it has the benefit of giving higher weighting to unfavorable conditions (observations further away from the mean), hence it is usually better at revealing model performance differences [3]. The criteria values (smaller-the-better) for each examined model are presented in Table 3. The proposed model (M23) was found to be superior in terms of prediction error (smaller RMSE value) and seems to provide overall the better fit (smaller AIC and second smaller BIC value) compared to all other models considered. Hence, the proposed model can be considered as an ideal founding mathematical model for early and accurate outbreak detection.

Plots of the time series, the predicted baseline level, and the threshold are illustrated in Figs. 3 and 4. The epidemics detected by the model (M23) appear in red in Fig. 3. The predicted baseline and threshold values at each date in

Table 1. Selected model M23 output

Parameter	Estimate	sd	t-value	p-value
α_0	34.137	1.888	18.084	<.001
α_1	−0.470	0.072	−6.511	<.001
α_2	0.003	0.001	5.167	<.001
γ_1	−14.030	0.837	−16.771	<.001
δ_1	19.678	0.855	23.002	<.001
γ_2	0.771	0.743	1.038	0.303
δ_2	−7.182	0.908	−7.907	<.001
γ_3	−0.858	0.764	−1.124	0.264
δ_3	3.153	0.770	4.096	<.001

Table 2. Models selected through the algorithm pathway

	Trend			Periodicity			Information criterion		R^2	
Model	t	t^2	t^3	1 year	6 months	3 months	AIC	BIC	Multiple R^2	Adjusted R^2
M11	*			*			610.47	622.91	0.799	0.793
M12	*			*	*		580.76	598.18	0.863	0.855
M13	*			*	*	*	571.51	593.91	0.882	0.872
M21	*	*		*			602.26	617.19	0.821	0.813
M22	*	*		*	*		561.57	581.48	0.892	0.884
M23	*	*		*	*	*	547.89	572.77	0.911	0.903
M31	*	*	*	*			603.21	620.63	0.823	0.813
M32	*	*	*	*	*		558.25	580.65	0.898	0.889
M33	*	*	*	*	*	*	546.79	574.17	0.914	0.904

Table 3. Comparative performance of forecasting models

Model	RMSE	AIC	BIC
M23	12.02	547.89	572.77
LT	48.34	818.44	823.75
MA3	13.51	554.72	565.33
SES	16.06	585.05	587.00
Holt's model	16.94	598.15	606.70
Brown's model	18.87	618.87	620.65
Winter's model	25.55	686.57	694.37
Serfling's model	18.81	610.47	622.91

the dataset are illustrated in Table 4 (see also Fig. 4). Detected epidemics also appear in the last column of Table 4 in bold. Table 5 presents the results of the retrospective evaluation of the excess influenza morbidity in Greece for 2014–2016, using the M23 periodic regression model. In particular, Table 5 shows the dates and excess morbidity for all detected epidemics. The excess morbidity is defined as the cumulative difference between observations and baseline over the entire epidemic period. Excess percentages are also provided in Table 5, calculated as the observed size divided by the sum of expected values throughout each epidemic.

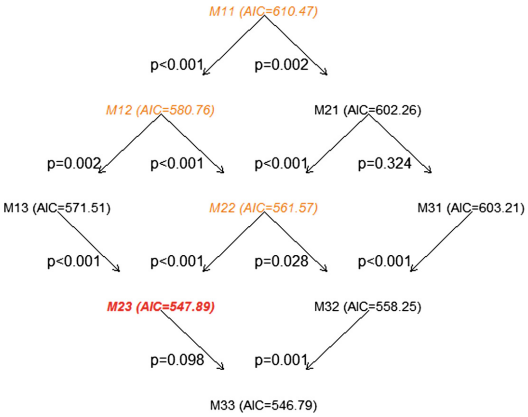


Fig. 1. Model selection pathway (ANOVA & AIC)

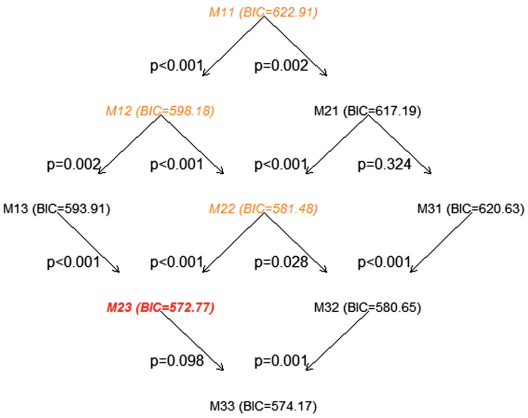


Fig. 2. Model selection pathway (ANOVA & BIC)

Table 4. Model results: Predicted baseline and epidemic threshold values

Week	Date[a]	PB[b]	TH[c]	Week	Date[a]	PB[b]	TH[c]
201440	2014-09-29	21.8	31.1	201441	2014-10-06	23.7	33.0
201442	2014-10-13	25.4	34.7	201443	2014-10-20	26.8	36.0
201444	2014-10-27	27.8	37.0	201445	2014-11-03	28.5	37.8
201446	2014-11-10	29.3	38.5	201447	2014-11-17	30.3	39.6
201448	2014-11-24	32.0	41.3	201449	2014-12-01	34.5	43.8
201450	2014-12-08	37.8	47.1	201451	2014-12-15	41.9	51.2
201452	2014-12-22	46.4	55.7	**201501**	**2014-12-29**	**50.9**	**60.1**
201502	**2015-01-05**	**54.8**	**64.1**	**201503**	**2015-01-12**	**57.9**	**67.1**
201504	**2015-01-19**	**59.7**	**68.9**	**201505**	**2015-01-26**	**60.1**	**69.3**
201506	**2015-02-02**	**59.1**	**68.4**	**201507**	**2015-02-09**	**57.0**	**66.3**
201508	**2015-02-16**	**54.2**	**63.4**	**201509**	**2015-02-23**	**50.9**	**60.1**
201510	**2015-03-02**	**47.5**	**56.8**	**201511**	**2015-03-09**	**44.2**	**53.5**
201512	**2015-03-16**	**41.2**	**50.5**	**201513**	**2015-03-23**	**38.3**	**47.6**
201514	2015-03-30	35.4	44.6	201515	2015-04-06	32.2	41.5
201516	2015-04-13	28.6	37.9	201517	2015-04-20	24.5	33.7
201518	2015-04-27	19.9	29.1	201519	2015-05-04	15.0	24.2
201520	2015-05-11	10.1	19.4	201521	2015-05-18	5.71	15.0
201522	2015-05-25	2.13	11.4	201523	2015-06-01	0.00	8.91
201524	2015-06-08	0.00	7.67	201525	2015-06-15	0.00	7.62
201526	2015-06-22	0.00	8.53	201527	2015-06-29	0.76	10.0
201528	2015-07-06	2.41	11.7	201529	2015-07-13	3.81	13.1
201530	2015-07-20	4.66	13.9	201531	2015-07-27	4.81	14.1
201532	2015-08-03	4.30	13.6	201533	2015-08-10	3.34	12.6
201534	2015-08-17	2.23	11.5	201535	2015-08-24	1.33	10.6
201536	2015-08-31	0.94	10.2	201537	2015-09-07	1.26	10.5
201538	2015-09-14	2.33	11.6	201539	2015-09-21	4.04	13.3
201540	2015-09-28	6.16	15.4	201541	2015-10-05	8.41	17.7
201542	2015-10-12	10.5	19.8	201543	2015-10-19	12.3	21.5
201544	2015-10-26	13.6	22.9	201545	2015-11-02	14.8	24.0
201546	2015-11-09	15.8	25.1	201547	2015-11-16	17.1	26.4
201548	2015-11-23	19.0	28.3	201549	2015-11-30	21.7	31.0
201550	2015-12-07	25.3	34.5	201551	2015-12-14	29.6	38.8
201552	2015-12-21	34.4	43.6	201553	2015-12-28	39.2	48.5
201601	**2016-01-04**	**43.6**	**52.9**	**201602**	**2016-01-11**	**47.2**	**56.5**
201603	**2016-01-18**	**49.6**	**58.8**	**201604**	**2016-01-25**	**50.6**	**59.8**
201605	**2016-02-01**	**50.2**	**59.5**	**201606**	**2016-02-08**	**48.6**	**57.9**
201607	**2016-02-15**	**46.2**	**55.5**	**201608**	**2016-02-22**	**43.3**	**52.6**
201609	2016-02-29	40.3	49.5	201610	2016-03-07	37.3	46.6
201611	2016-03-14	34.6	43.8	201612	2016-03-21	32.0	41.3
201613	2016-03-28	29.5	38.7	201614	2016-04-04	26.7	35.9
201615	2016-04-11	23.5	32.8	201616	2016-04-18	19.8	29.1
201617	2016-04-25	15.6	24.9	201618	2016-05-02	11.1	20.4
201619	2016-05-09	6.56	15.8	201620	2016-05-16	2.39	11.6
201621	2016-05-23	0.00	8.23	201622	2016-05-30	0.00	5.88
201623	2016-06-06	0.00	4.76	201624	2016-06-13	0.00	4.85
201625	2016-06-20	0.00	5.95	201626	2016-06-27	0.00	7.72
201627	2016-07-04	0.46	9.71	201628	2016-07-11	2.27	11.5
201629	2016-07-18	3.57	12.8	201630	2016-07-25	4.19	13.4
201631	2016-08-01	4.13	13.4	201632	2016-08-08	3.57	12.8
201633	2016-08-15	2.80	12.1	201634	2016-08-22	2.18	11.4
201635	2016-08-29	2.03	11.3	201636	2016-09-05	2.56	11.8
201637	2016-09-12	3.84	13.1	201638	2016-09-19	5.79	15.0
201639	2016-09-26	8.20	17.5				

[a] Date denotes the week start date;
[b] PB denotes the predicted baseline;
[c] TH denotes the threshold.

Table 5. Retrospective evaluation of the excess influenza morbidity in Greece 2014–2016

SW [a]	EW [a]	Excess cases	Expected cases	Cases	Excess percentage
201501	201513	523	676	1199	77%
201601	201608	222	379	601	58%

[a]SW and EW denote the signaled start and end weeks for epidemics, respectively.

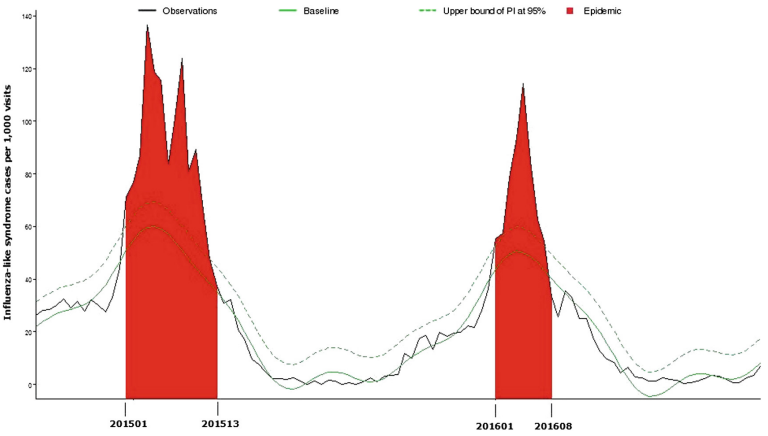


Fig. 3. Detected epidemics in Greece 2014–2016 (Color figure online)

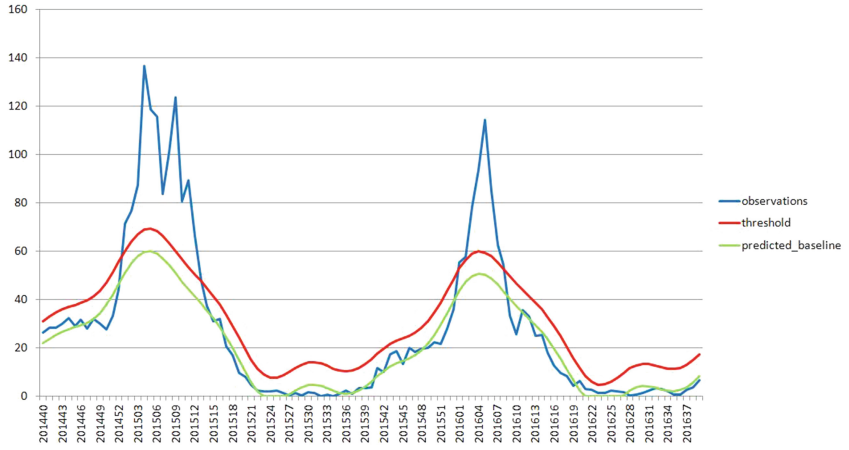


Fig. 4. Weekly influenza morbidity in Greece 2014–2016

4 Concluding Remarks

In this paper, we conducted a retrospective analysis for the estimation of the influenza-like syndrome morbidity burden in Greece for the period 2014–2016 (week40/2014 to week39/2016). The model was fitted over all data to estimate the baseline level and the epidemic threshold for the whole observation period, allowing us to extract the signaled start (sw) and end weeks (ew) of the epidemics (i.e., sw01-ew13/2015, sw01-ew08/2016), and to measure the excess during these epidemic periods. Periodic regression models were used to estimate an expected baseline level for the time series, associated with a prediction interval. In this way, the dates of epidemics were extracted, the related morbidity burden was estimated, and alert thresholds could be used to perform real time surveillance of influenza-like syndrome. The periodic regression model selected as the optimal one, succeeded in detecting the beginning and end of the epidemic wave for both periods (2014–2015, 2015–2016), and therefore identified the pattern that best matches the recent influenza activity. It is worth noting that it also succeeded in detecting as epidemic the period between the two peaks of the epidemic wave for the period of 2014–2015. Moreover, the selected model (M23) compared to typical trend or/and seasonality detection models was found to perform favorably.

Within this framework, the present work provided general recommendations to serve critical needs of Public Health for the very early and accurate detection of epidemic activity by conducting a comparative experimental real data study. The interconnection of statistical research with Health professional's structures ensures the scientific merit of the findings from a statistical as well as an epidemiological perspective. This interaction of scientific areas and ideas will benefit all stakeholders with the ultimate aim of exploitation and application of the scientific findings in clinical and public health practice.

Acknowledgments. The postdoctoral research of the first author is financially supported by a postdoc scholarship awarded by "IKY FELLOWSHIPS OF EXCELLENCE FOR POSTGRADUATE STUDIES IN GREECE-SIEMENS PROGRAM". The supervision of the postdoctoral research work is carried out by the second author. The work was carried out at the Lab of Statistics and Data Analysis of the University of the Aegean. The authors would like to thank the Department of Epidemiological Surveillance and Intervention of the Hellenic Center for Disease Control and Prevention for providing the influenza-like illness (ILI) rate data, collected weekly through the sentinel surveillance system.

References

1. Brillman, J.C., Burr, T., Forslund, D., Joyce, E., Picard, R., Umland, E.: Modeling emergency department visit patterns for infectious disease complaints: results and application to disease surveillance. BMC Med. Inform. Decis. Mak. **5**(1), 4 (2005)
2. Buckeridge, D.L., Burkom, H.S., Campell, M., Hogan, W.R., Moore, A.: Algorithms for rapid outbreak detection: a research synthesis. J. Biomed. Inform. **38**, 99–113 (2005)

3. Chai, T., Draxler, R.R.: Root mean square error (RMSE) or mean absolute error (MAE)?-Arguments against avoiding RMSE in the literature. Geosci. Model Dev. **7**, 1247–1250 (2014)
4. Costagliola, D., Flahault, A., Galinec, D., Garnerin, P., Menares, J., Valleron, A.-J.: When is the epidemic warning cut-off point exceeded? Eur. J. Epidemiol. **10**, 475–476 (1994)
5. Costagliola, D., Flahault, A., Galinec, D., Garnerin, P., Menares, J., Valleron, A.-J.: A routine tool for detection and assessment of epidemics of influenza-like syndromes in France. Am. J. Public Health **81**, 97–99 (1991)
6. Farrington, C.P., Andrews, N.: Outbreak detection: Application to infectious disease surveillance. In: Brookmeyer, R., Stroup, D.F. (eds.) Monitoring the Health of Populations: Statistical Principles & Methods for Public Health Surveillance, pp. 203–231. Oxford University Press, Oxford (2004)
7. Fleming, D.M., Rotar-Pavlic, D.: Information from primary care: its importance and value. A comparison of information from Slovenia and England and Wales, viewed from the "Health 21" perspective. Eur. J. Public Health **12**, 249–253 (2002)
8. Heath, I., Smeeth, L.: Tackling health inequalities in primary care [Editorial]. Br. Med. J. **318**, 1020–1021 (1999)
9. Housworth, J., Langmuir, A.D.: Excess mortality from epidemic influenza, 1957–1966. Am. J. Epidemiol. **100**, 40–48 (1974)
10. Hulth, A., Andrews, N., Ethelberg, S., Dreesman, J., Faensen, D., van Pelt, W., Schnitzler, J.: Practical usage of computer-supported outbreak detection in five European countries. Eurosuveillance **15**(36), 1–6 (2010). pii=19658
11. Langmuir, A.D.: The surveillance of communicable diseases of national importance. In: Buck, C., Liopis, A., Najera, E., Terris, M., (eds.) The Challenge of Epidemiology. Issues and Selected Readings, 3rd edn., pp. 855–867. Pan American Health Organisation, Washington, DC (1995)
12. Last, J.M.: A Dictionary of Epidemiology. Oxford University Press, Oxford (1995)
13. Lui, K.J., Kendal, A.P.: Impact of influenza epidemics on mortality in the United States from October 1972 to May 1985. Am. J. Public Health **77**, 712–716 (1987)
14. Macmahon, B., Trichopoulos, D.: Epidemiology: Principles and Methods. Little, Brown and Co., Boston (1996)
15. Norbury, M., Mercer, S.W., Gillies, J., Furler, J., Watt, G.C.M.: Time to care: tackling health inequalities through primary care. Fam. Pract. **28**, 1–3 (2011)
16. Olson, D.R., Simonsen, L., Edelson, P.J., Morse, S.S.: Epidemiological evidence of an early wave of the 1918 influenza pandemic in New York City. Proc. Natl. Acad. Sci. USA **102**, 11059–11063 (2005)
17. Pelat, C., Boëlle, P.-Y., Cowling, B.J., Carrat, F., Flahault, A., Ansart, S., Valleron, A.-J.: Online detection and quantification of epidemics. BMC Med. Inform. Decis. Mak. **7**, 29 (2007). https://doi.org/10.1186/1472-6947-7-29
18. Sebastiani, P., Mandl, K.: Biosurveillance and outbreak detection. In: Data Mining: Next Generation Challenges and Future Directions, pp. 185–198 (2004)
19. Serfling, R.: Methods for current statistical analysis of excess pneumonia-influenza deaths. Public Health Rep. **78**, 494–506 (1963)
20. Shmueli, G., Burkom, H.: Statistical challenges facing early outbreak detection in biosurveillance. Technometrics **52**, 39–51 (2010)
21. Simonsen, L., Reichert, T.A., Viboud, C., Blackwelder, W.C., Taylor, R.J., Miller, M.A.: Impact of influenza vaccination on seasonal mortality in the US elderly population. Arch. Intern. Med. **165**, 265–272 (2005)
22. Sonesson, C., Bock, D.: A review and discussion of prospective statistical surveillance in public health. J. Royal Stat. Soc. Ser. A **166**, 5–21 (2003)

23. Starfield, B., Shi, L., Mackinko, J.: Contribution of primary care to health systems and health. Milbank Q. **83**, 457–502 (2005)
24. Stroup, D.F., Williamson, G.D., Herndon, J.L., Karon, J.M.: Detection of aberrations in the occurence of notifiable diseases surveillance data. Stat. Med. **8**, 323–329 (1989)
25. Teutsch, S.M., Thacker, S.B.: Planning a public health surveillance system. Epidemiol. Bull. **16**, 1–6 (1995)
26. Thompson, W.W., Shay, D.K., Weintraub, E., Brammer, L., Cox, N., Anderson, L.J., Fukuda, K.: Mortality associated with influenza and respiratory syncytial virus in the United States. JAMA **289**, 179–186 (2003)
27. Unkel, S., Farrington, C.P., Garthwaite, P.H., Robertson, C., Andrews, N.: Statistical methods for the prospective detection of infectious disease outbreaks: a review. J. Royal Stat. Soc. Ser. A **175**, 49–82 (2012)
28. Vellinga, A., Van Loock, F.: The dioxin crisis as experiment to determine poultry-related campylobacter enteritis. Emerg. Infect. Dis. **8**, 19–22 (2002)
29. Vergu, E., Grais, R.F., Sarter, H., Fagot, J.P., Lambert, B., Valleron, A.J., Flahault, A.: Medication sales and syndromic surveillance, France. Emerg. Infect. Dis. **12**, 416–421 (2006)
30. Viboud, C., Boelle, P.Y., Pakdaman, K., Carrat, F., Valleron, A.J., Flahault, A.: Influenza epidemics in the United States, France, and Australia, 1972–1997. Emerg. Infect. Dis. **10**, 32–39 (2004)
31. World Health Organization (2017). http://www.who.int/topics/public_health_surveillance/en/

Mining Acute Stroke Patients' Data Using Supervised Machine Learning

Ritu Kundu(✉) and Toktam Mahmoodi

Department of Informatics, King's College London, London WC2R 2LS, UK
{ritu.kundu,toktam.mahmoodi}@kcl.ac.uk

Abstract. Analysis of data for identifying patterns and building models has been used as a strong tool in different domains, including medical domains. In this paper, we analyse the registry of brain stroke patients collected over fifteen years in south London hospitals, known as South London Stroke Register. Our attempt is to identify the similar patterns between patients' background and living conditions, their cognitive ability, the treatments they received, and the speed of their cognitive recovery; based on which most effective treatment can be predicted for new admitted patients. We designed a novel strategy which takes into account two different approaches. First is to predict, for each of the potential intervention treatments, whether that particular treatment would lead to recovery of a new patient or not. Second is to suggest a treatment (treatments) for the patient based on those that were given to the patients who have recovered and are most similar to the new patient. We built different classifiers using various state of the art machine learning algorithms. These algorithms were evaluated and compared based on three performance metrics, defined in this paper. Given that time is very crucial for stroke patients, main motivation of this research work is identifying the most effective treatment immediately for a new patient, and potentially increase the probability of their cognitive recovery.

Keywords: Data mining · Modelling and analysis of clinical data
Machine learning algorithms

1 Introduction

Brain stroke is one of the major health concerns. According to the statistics [1], in 2010 stroke was the fourth largest cause of death in the UK after cancer, heart disease and respiratory disease; causing almost 50,000 deaths. Further, more than half of all stroke survivors are left dependent on others for everyday activities. There are approximately 152,000 brain strokes reported in the UK every year, and it is a leading cause of adult disability. According to a recent report, there is an alarming increase in the numbers of people having a stroke in working age [2].

It is important to treat stroke patients immediately with the most efficient treatment. Understanding how recovery and treatments are influenced by

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 364–377, 2017.
https://doi.org/10.1007/978-3-319-72453-9_30

patients' *individual* extent of injury (brain damage), and their socio-demographic and medical background could results in more faster and more effective treatments. In other words, individual stroke treatment decision making is more probable to be successful. Prognosis for recovery of an acute stroke patient given a particular intervention treatment can aid the decision making by healthcare professionals. Moreover, suggestions of potentially effective treatments for a new patient based on the other patients with similar clinical, medical and socio-demographical factors who have recovered in the past, can be a guide to decide an appropriate and reliable treatment approach. Prognosis can be done using a model that is based on classifiers which use a set of pre-treatment assessment variables for prediction (classification). These classifiers can be built using machine learning techniques as an alternative to the usual approach of analysing the stroke-data through logistic regression models. Machine learning approach allows exploration of the data leading to interesting, previously unknown, patterns being revealed. Additionally, the greatest strength of machine learning techniques lies in their potential to improve performance by easily incorporating newly available data [24].

Presented here, is an observational study that explores the possibility of using various machine learning techniques to build a tool for assisting medical experts in selecting the most effective intervention-treatment approach. For cases such as stroke, there is a very small window of time, during which treatments can be the most effective and hence a fast and accurate choice of treatment can significantly increases the chances of recovery for a patient after an acute stroke. More specifically, we study various machine-learning algorithms that can be used to train distinct classifiers which can, later, be combined into one software tool, or application, to be used by medical experts.

The paper is organised as follows: The next section provides a short literature review and specifies the contribution of this study. The following section gives an insight into the data used for this study, followed by Sect. 4 which explains the details of the methodology adopted. After that, Sect. 5 covering the aspects of modelling the data follows. Results obtained are presented and discussed in Sect. 6. Section 7 concludes the study presented.

2 Related Work

Thus far, there have been a few studies adopting machine learning techniques for analysing the stroke-data. Recently, a study utilizes one machine learning technique, support vector machine (SVM), on computerized tomography (CT) images along with clinical variables, for prediction of symptomatic intracranial haemorrhage (SICH) associated with intravenous thrombolysis administered to acute ischemic stroke patients [6]. Another recent study applies machine learning algorithms in acute ischemic stroke outcome prediction in relation to treatment by endovascular intervention [4]. Another study, using MRI of rats, compares five predictive algorithms (generalized linear model (GLM), generalized additive model, support vector machine, adaptive boosting, and random forest) to predict

brain infarction, and differentiate potentially salvageable tissue from irreversibly damaged tissue [7]. Yet another study uses spatially regularized SVM on brain images of acute stroke patients to detect brain areas associated with motor outcome at 90 days, based on diffusion-weighted images acquired at the acute stage [9].

Our Contribution. There are numerous instances of the research work done earlier which employ machine learning techniques for predicting the outcomes of the stroke treatment. However, the prognostic models designed in these studies only focus on whether administering a particular treatment will be beneficial; moreover, only one machine learning method (SVM) is mostly used.

To the best of our knowledge, no attempts have been made to apply a broad range of machine learning algorithms for building a *comprehensive multifactorial model* which can predict the outcome of each of the usual intervention-treatment approaches and can suggest, in addition, potentially more promising treatments based on the *similarity of the patient* to the patients who have recovered. More specifically, this is the first study which incorporates the following:

1. Applying a range of machine learning algorithms for building a comprehensive multifactorial model which can predict the outcome of each of the usual intervention-treatment approaches.
2. Devicing a strategy that is working on two horizons:
   - First Approach: Predicting whether a patient would recover or not if a particular treatment is used, for each of the possible treatments.
   - Second Approach: Suggesting the subtype of each treatment-type based on similarity of the patient with recovered patients.

3 Dataset

The dataset used in the study is a sample set obtained from the community-based South London Stroke Register (SLSR)[1]. The SLSR is a prospective population-based stroke register set up in January 1995, recording all first-ever strokes in patients of all ages for an inner area of South London based on 22 electoral wards in Lambeth and Southwark, over 20 years. In the data-set being used, patients were assessed for cognitive function using Abbreviated Mental Test [11] or Mini-Mental State Examination [15] at the onset, 3 months, and annually thereafter. In addition, various details related to socio-demographics, various risk factors prior to stroke, previous medical history, stroke symptoms and the severity of stroke are noted and taken into account to decide for the acute intervention methods. Later, various medical tests like ECG, ECHO, blood investigations and brain imaging are used for stroke classification.

[1] details can be found online at: KCL Faculty of Life Sciences and Medicine, Stroke research group.

4 Methodology

Our methodology to study the SLSR dataset comprised of three steps, as follows:

1. We designed the strategy for addressing the problem, i.e. defined the criteria to build classifiers and identified the classes to train classifiers.
2. Data was pre-processed which included classifying the data, cleaning up the data, bringing it in the form required by the classification techniques and splitting the dataset to obtain the required subsets.
3. Data was modelled by training classifiers on the pre-processed data using different classification algorithms. Training was followed by comparing the performance of the different classification algorithms to study the pros and cons of various classifiers with respect to their application to the SLSR dataset.

The first two steps have been described in the following subsections while the third step has been elaborated in Sect. 5.

4.1 Defining Classifications Criteria

Figure 1 demonstrates different types (classes) of treatments and the sub-types under each main class. The solution was designed using two different strategies for developing a prediction-model that could assist in choosing the most promising intervention-treatment.

- **First Approach: Predicting whether a patient would recover or not if a particular treatment is used.**
  In this approach each sub-type of the main treatment types was viewed as representation of a *treatment-class*. For each type of the possible treatment-classes, a classifier was built to predict whether that particular treatment-type would result in a patient being recovered or not.
- **Second Approach: Suggesting the subtype of each treatment-type based on similar records of recovered patients.**
  Each record of a recovered patient, in this approach, was seen as a point in a space with number of dimensions equal to that of predictive-attributes being considered. For prediction, similarity of the new record (new patient) from the other records (patients who recovered in the past) was calculated and 'k' (a pre-defined positive constant number) nearest neighbours were decided to be considered. Class of the majority of these k nearest neighbours, i.e. treatment given to the majority of these *similar patients*, would be assigned to the new record. Thus, for each treatment type, its subtype could be suggested using this approach.

4.2 Pre-processing Data

For training classifiers, instances (records) had to be dichotomised in classes corresponding to recovered and not-recovered patients. It was decided after a

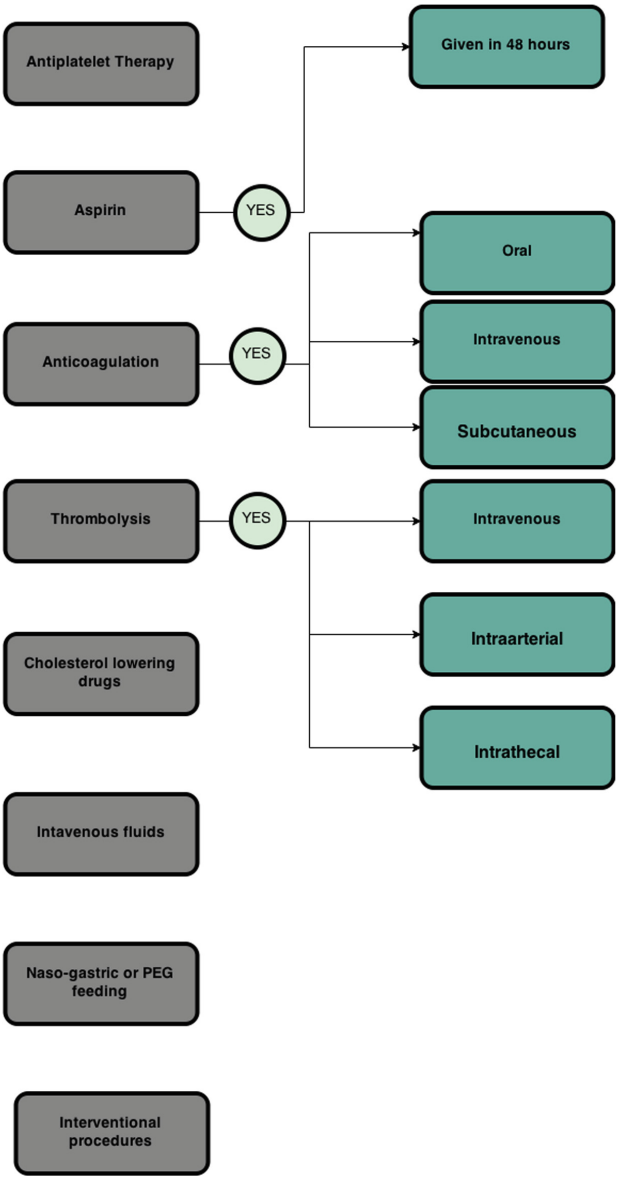


Fig. 1. Different types of acute intervention treatments and their sub-types

discussion with a medical expert that the recovery status could be determined by the scores used to access cognitive impairment. In the given dataset, following scores were recorded:

Table 1. Different types of treatments considered in the classifiers.

Treatment index	Type of treatment
1	Antiplatelet therapy (marked as Antiplatelet)
2	Aspirin
3	Anticoagulation-subcutaneous (marked as Anticoag subcutaneous)
4	Anticoagulation-oral (marked as Anticoagulation oral)
5	Thrombolysis-oral (marked as Thrombo oral)
6	Cholesterol lowering drugs (marked as Cholesterol)
7	Naso-gastric or PEG feeding (marked as NP feeding)
8	Intavenous fluids (marked as Intavenous)

- AMT score (recorded 1 January, 2000 onwards) : Abbreviated mental test score is assigned on a scale of 10. 7 is used as the threshold, i.e. score less than or equal to 7 implies cognitive impairment [25].
- MMSE score (recorded before 1 January, 2000): Mini-mental state examination score is rated on a scale of 30 and 24 is considered as the threshold [25].

Since AMT score was observed in the follow-ups as well, an appropriate criteria seemed to label a record as *recovered* where AMT score was above the threshold for every observation and tag a record as *not-recovered* if AMT score was below the threshold for each observation. In the cases where MMSE score was noted initially, it was scaled accordingly to reflect a corresponding value on AMT scale of 10. Since there was no clear monotonous trend in the observations, moving average technique was used to smooth out the short-term fluctuations and capture a long-term stable trend. A window of 3 was taken for the running average.

After labelling, cleaning-up was performed followed by categorisation of the numeric values so as to convert them to nominal (as required by many classifier-training algorithms) and removal of unnecessary attributes. We obtained 520 labelled records in total after labelling and cleaning-up.

Next, data was split as follows: For the first approach, data ($n = 520$) was split along the lines of the treatment-classes, i.e. data-records corresponding to the patients given that treatment were separated from the rest. These subsets of the data were then used (one at a time) to train different classifiers for classifying new records into classes corresponding to 'recovered-patients' and 'not-recovered-patient'. All the treatments listed in Table 1 had been considered for the experiment as they had sufficient data to result in an effective classifier.

For the second approach, only the records labelled as 'recovered' were considered ($n = 390$). Subsets of the data had been generated such that the considered category of treatments were non-empty. Therefore, the first seven categories of Table 1 were considered while lines 3 and 4 of the table were considered as a single category of "AntiCogulation". This completes the pre-processing stage of our methodology.

5 Modelling Data

For modelling SLSR data, we used a number of supervised-learning algorithms representing different approaches of machine learning and these also have been used widely in data-mining studies done so far on medical data. After training the classifiers, their performance in modelling the data was examined and compared.

5.1 Software Tool

The software tool used for training the classifiers as well as for evaluating their performance is *Weka* [13]. Weka (Waikato Environment for Knowledge Analysis), developed by the University of Waikato, is a cross-platform open source and one of the most popular software for machine learning based applications. It has been written in Java and contains a collection of visualization tools and algorithms for data-mining tasks such as data analysis and predictive modelling[2].

5.2 Classification Techniques

In principle, any of the machine learning algorithms can be used to train a classifier, but each of them has its own benefits and limitations depending on the type of data it is being applied. Following algorithms were employed based on their default parameters of the WEKA application on the subsets of the data obtained by splitting the pre-processed and cleaned data.

First Approach

1. Naive Bayes classifier: Probabilistic classifier based on Bayes Theorem. For each class value, it predicts the probability that a given instance belongs to that class. The class having the highest probability is assigned to that instance [16].
2. Support Vector Machine (SVM): It is hyper-plane classifier based on margin maximisation between the target classes by mapping input space to higher dimensional space and thus achieving linear separability [17].
3. Multilayer Perceptron (MLP): A feed-forward back-propagation network consisting of input, output, and one or more hidden layers. It extracts useful information while learning to assign weighted coefficients to components of the input layer [22].
4. Conjective Rule based classifier: It is based on a decision-making rule that uses AND logical relation to correlate stimulus attributes. This rule consists of antecedents (attributes) "AND" ed together and the class value for the classification. It uses *Information Gain* to select the antecedent [14].
5. Decision Tables based classifier: It is also a rule-based classifier that builds a decision table based on the labelled instances [18].

[2] For this study Weka 3.6.11 has been used.

6. Alternating Decision Tree: It is a generalisation of decision tree, voted decision tree, and voted decision stumps. It has two types of nodes - decision node (containing a predicate) and prediction node (containing a single number). It is different from the other decision trees such as C4.5 in which an instance travels only a single path through the tree. In ADTree, an instance follows all the paths for which decision nodes are true, summing up any predicate nodes traversed [12].
7. C4.8: It is an extension of Quinlan's earlier ID3 algorithm. It produces a decision tree from labelled instances using the concept of information entropy [21]. It is 'J48' in Weka.
8. Naive Bayes classifier based Decision Tree : It is a hybrid of decision tree and naive Bayes classifiers. It produces a decision tree with naive Bayes classifiers at the leaves [20].

Second Approach

1. K-nearest neighbours classifier(KNN): It is an instance based classifier that uses similarity of a given instance with other instances to choose its neighbours and uses the majority of neighbours to classify that instance. K is a positive integer [3]. It is same as 'IBk' in Weka.
2. KStar (K*): It is also an instance-based classifier similar to KNN classifier but it uses entropy as the distance measure [8].

The value of $k = 3$ has been used for the above two algorithms.

5.3 Testing Technique

10-fold Cross Validation [23] was used as the testing technique to minimize the bias associated with random sampling of training and test data samples [19].

5.4 Performance Metrics for Evaluation

We have evaluated different classification techniques using three performance indicators: Prediction accuracy, Kappa measure, and Area under Receiver Operating Characteristics (ROC) curve.

The prediction accuracy was computed based on the proportion of instances classified correctly. In this study, accuracy was calculated by averaging the results from 10 runs of 10-fold cross validation. The second performance metric chosen was Kappa measure because Cohen's Kappa statistic is mostly in agreement with the overall accuracy but proves to be better in case of unbalanced datasets as it compensates for the classification that may be due to chance [5]. The third metric selected was Area under ROC Curve (AUC). A Receiver Operating Characteristics (ROC) graph is a technique for selecting classifiers based on their performance. It is one of the most commonly used evaluation criteria of classifiers in medical-domain. Area under the two-dimensional ROC curve is a well-used method for comparing classifiers [10].

6 Evaluation Results and Comparison

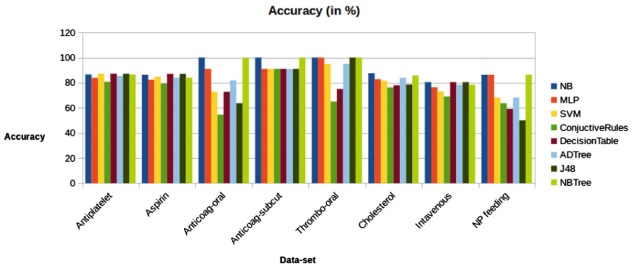
Figures 2a, b, and c demonstrate evaluation results from the first approach, corresponding to the chosen performance metrics (Accuracy, Kappa measure and AUC). Evaluation results obtained for the second approach are illustrated in the Figs. 3a, b, and c. All of these figures also compare different algorithms used for training each of the classifiers (corresponding to every data subset). These results achieved by evaluating various classifiers are analysed and discussed in the following subsections.

6.1 First Approach

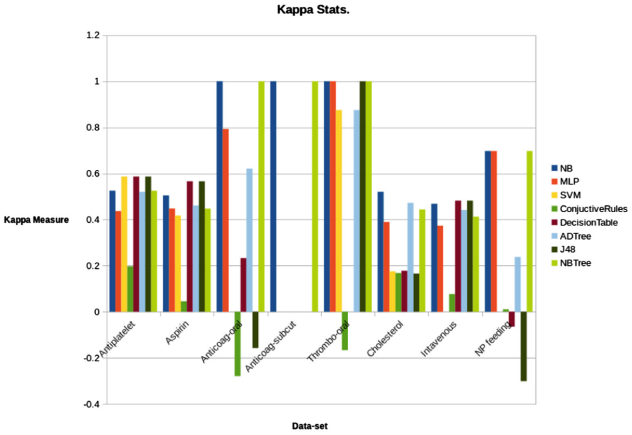
We summarize the results from the first approach (Fig. 2) in Table 2, in which the best performing algorithms based on each of the performance-metrics on each of the data-subsets can be seen. Overall, NB, MLP and tree-based algorithms are performing quite well, in comparison with the rule-based algorithms. Additionally, it can be seen that Accuracy and Kappa measure are in agreement with each other, but AUC has different behaviour for a particular dataset. Moreover, it can be observed that SVM is performing worst on all of the datasets according to all three comparison-parameters.

Table 2. First approach: Best performing classification algorithms for each of the data-subsets

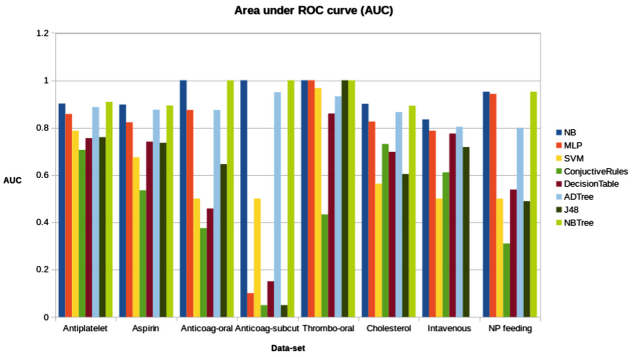
	Accuracy (in %)	Kappa measure	Area under ROC
Antiplatelet	J48	J48	ADTree
	87.0968	0.578	0.871
Aspirin	ADTree	ADTree	NB
	85.7988	0.5375	0.846
Anticoag-subcut	All except SVM	NB, NBTree	NB, NBTree
	90.9091	0.6207	1
Anticoag-oral	MLP, ConjuctiveRules, ADTree, J48	NB, DecisionTable, NBTree	ADTree
	72.7273	0.2326	0.75
Thrombo-oral	NB, J48, NBTree	J48	NB, J48, NBTree
	90	0.76447	1
Cholesterol	ADTree	ADTree	ADTree, J48
	86.3095	0.4889	0.883
Intavenous	ADTree	ADTree	ADTree
	79.3814	0.4476	0.807
NP feeding	MLP	MLP	NB, NBTree
	81.8182	0.581	0.867



(a)

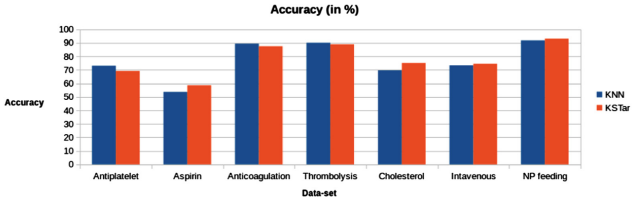


(b)

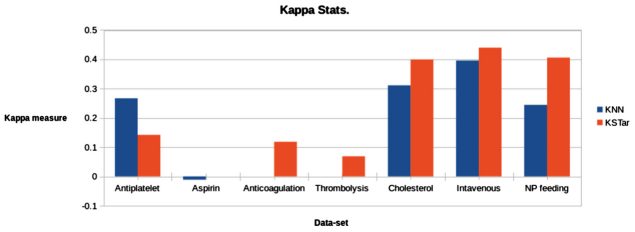


(c)

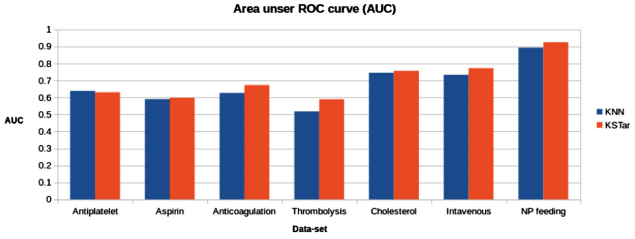
Fig. 2. First approach: Comparison of classifiers' (a) accuracy (b) Kappa measure (c) AUC.



(a)



(b)



(c)

Fig. 3. Second approach: Comparison of classifiers' (a) accuracy (b) Kappa measure (c) AUC.

6.2 Second Approach

As can be seen from Table 3, both of the chosen algorithms are performing bad on the basis of Kappa measure. However, based on the other two metrics, it can be inferred that KStar is performing better than KNN for the first four datasets while KNN is the clear winner for the remainder of three datasets. In addition, it seems that all the three comparison parameters are approximately in agreement with each other for this approach, which is in contrast to the results obtained for the first approach.

Table 3. Second approach: Best performing classification algorithms for each of the data-subsets

	Accuracy (in %)	Kappa measure	Area under ROC
Antiplatelet	Both	KStar	KStar
	65.019	0.168	0.612
Aspirin	KStar	KStar	KStar
	54.5455	0.2287	0.634
Anticoagulation	KNN	KStar	KStar
	88.9734	0.0504	0.565
Thrombolysis	KNN	KNN	KStar
	88.5496	0.0041	0.562
Cholesterol-oral	KNN	KNN	KNN
	64.9805	0.1647	0.68
Intavenous	KNN	KNN	KNN
	67.6113	0.2676	0.709
NP feeding	KNN	KStar	KNN
	91.8605	0.179	0.82

7 Conclusion

In this paper, we applied data mining methods, i.e. machine learning, to the historical data from stroke patients obtained from SLSR. This dataset recorded various details of the patients related to socio-demographics, various risk factors prior to stroke, previous medical history, stroke symptoms, and the severity of stroke. We designed a novel strategy which takes into account two different approaches. First, to predict, for each of the potential intervention treatments, whether that particular treatment would lead to recovery of a new patient or not. Second, to suggest treatment(s) for the new patient, based on those which were given to the patients in the past, with similar profile to the new patient, and were successful. We built different classifiers using various machine learning algorithms. These algorithms were evaluated and compared based on three performance metrics: Accuracy, Kappa measure, and Area under ROC. Comparison of the algorithms for each of the classifiers let us gain a clearer insight into which classification algorithm would work better on which of the classifier (subset of the dataset).

Given that most of the existing studies on medical data using machine learning techniques are limited in terms of focussing on prediction of the outcome after stroke for a particular treatment, there is a need for more thorough models. Our proposed model, therefore, covers all the possible treatment options at the same time and results in more accurate analysis, if used with a sufficiently large dataset. The outcome of our model can be used in clinical decision making

to significantly increase the probability of recovery by choosing the appropriate intervention treatment.

The ultimate, more usable, outcome of this work can be in the form of a user friendly software that combines different classifiers from both the approaches which are analysed here, for making the treatment suggestions. This software application can be used by medical experts to assist them in quickly choosing the most promising treatment so that the chances of recovery of the patient increases.

Acknowledgement. The authors would like to thank Miss Siobhan Crichton from the department of Primary Care & Public Health Sciences at King's College for providing support on the dataset, to the authors. Also, we would like to thank Dr. Arshia Sedigh at King's College Hospital for his valuable advice on understanding the data without which this work was not possible.

References

1. State of the Nation: stroke statistics. Stroke Association, January 2015. http://www.stroke.org.uk/
2. Strokes rising among people of working age, warns charity. BBC Health news, March 2015. http://www.bbc.com/news/health-32690040
3. Aha, D., Kibler, D.: Instance-based learning algorithms. Mach. Learn. **6**, 37–66 (1991)
4. Asadi, H., Dowling, R., Yan, B., Mitchell, P.: Machine learning for outcome prediction of acute ischemic stroke post intra-arterial therapy. PLoS ONE 9 (2014). https://doi.org/10.1371/journal.pone.0088225
5. Ben-David, A.: Comparison of classification accuracy using cohen's weighted kappa. Expert Syst. Appl. **34**(2), 82–832 (2009)
6. Bentley, P., Ganesalingam, J., Jones, A.L.C., Mahady, K., Epton, S., Rinne, P., Sharma, P., Halse, O., Mehta, A., Rueckert, D.: Prediction of stroke thrombolysis outcome using CT brain machine learning. NeuroImage Clin. **4**, 635–640 (2014)
7. Bouts, M.J., Tiebosch, I.A., van der Toorn, A., Viergever, M.A., Wu, O., Dijkhuizen, R.M.: Early identification of potentially salvageable tissue with MRI-based predictive algorithms after experimental ischemic stroke. J. Cereb. Blood Flow Metab. **33**(7), 1075–1082 (2013)
8. Cleary, J.G., Trigg, L.E.: K*: An instance-based learner using an entropic distance measure. In: 12th International Conference on Machine Learning, pp. 108–114 (1995)
9. Cuingnet, R., Rosso, C., Lehéricy, S., Dormont, D., Benali, H., Samson, Y., Colliot, O.: Spatially regularized SVM for the detection of brain areas associated with stroke outcome. In: Jiang, T., Navab, N., Pluim, J.P.W., Viergever, M.A. (eds.) MICCAI 2010. LNCS, vol. 6361, pp. 316–323. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15705-9_39
10. Fawcett, T.: An introduction to ROC analysis. Pattern Recogn. Lett. **14**(27), 861–874 (2006)
11. Folstein, M.F., Folstein, S.E., McHugh, P.R.: Mini-mental state: a practical method for grading the cognitive state of patients for the clinician. J. Psychiatr. Res. **12**(3), 189–198 (1975)

12. Freund, Y., Mason, L.: The alternating decision tree learning algorithm. In: Proceeding of the Sixteenth International Conference on Machine Learning, Bled, Slovenia, pp. 124–133 (1999)
13. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explor. **11**(1), 385–403 (2009)
14. Har-Peled S., Roth, D., Zimak, D.: Constraint classification for multiclass classification and ranking. In: 16th Annual Conference on Neural Information Processing Systems, NIPS-02, pp. 785–792. MIT Press (2003)
15. Hodkinson, H.M.: Evaluation of a mental test score for assessment of mental impairment in the elderly. Age Ageing **1**(4), 233–238 (1972)
16. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 338–345. Morgan Kaufmann, San Mateo (1995)
17. Kecman, V.: Learning and Soft Computing: Support Vector Machines, Neural Networks and Fuzzy Logic Systems. MIT Press, Cambridge (2001)
18. Kohavi, R.: The power of decision tables. In: Lavrac, N., Wrobel, S. (eds.) ECML 1995. LNCS, vol. 912, pp. 174–189. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59286-5_57
19. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceeding of 14th International Joint Conference on Artificial Intelligence, (IJCAI 1995), pp. 1137–1143 (1995)
20. Kohavi, R.: Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In: Second International Conference on Knowledge Discovery and Data Mining, pp. 202–207 (1996)
21. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
22. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, pp. 318–362. MIT Press (1986)
23. Varma, S., Simon, R.: Bias in error estimation when using cross-validation for model selection. BMC Bioinformatics 7(91) (2006)
24. Warwick, K.: March of the Machines: The Breakthrough in Artificial Intelligence. University of Illinois Press, Champaign (2004)
25. Wolfe, C.D., Crichton, S.L., Heuschmann, P.U., McKevitt, C.J., Toschke, A.M., Grieve, A.P., Rudd, A.G.: Estimates of outcomes up to ten years after stroke: analysis from the prospective South London stroke register. PLoS Med. 8(5) (2011)

Parallel and Robust Empirical Risk Minimization via the Median Trick

Alexander Kogler and Patrick Traxler(✉)

Data Analysis Systems Group, Software Competence Center Hagenberg,
Hagenberg im Mühlkreis, Austria
{alexander.kogler,patrick.traxler}@scch.at

Abstract. The median trick is a technique to boost the success probability of algorithms. We apply it to empirical risk minimization (ERM) and related problems. We obtain a *parallel ERM principle*, i.e. we get parallel, scalable algorithms for many learning problems. We provide generalization bounds and carry out computer experiments to demonstrate the practical effectiveness of the median trick. Our results can be summarized as follows: The median trick applies to a large class of classification and regression problems. It is simple to implement, scales well, and is robust due to the application of the median. The trade-off is a slightly decreased accuracy compared to sequential algorithms.

Keywords: Empirical risk minimization · Parallel algorithms
Median trick · Generalization bounds · Classification · Regression

1 Introduction

Parallel and distributed learning algorithms are increasingly important due to the need for analyzing large data sets. A common approach to analyze mid-sized to large data sets is to employ parallel and distributed algorithms for solving the underlying optimization problems. In this work we pursue a different approach. We present a technique to parallelize a given sequential learning algorithm. The technique is based on the median trick or median-of-means. It can be seen as meta-algorithm, see Algorithm 1, that solves, say, a binary classification via hinge loss optimization by partitioning the data into groups of equal size. It then solves the hinge loss optimization problem on every group in parallel. The subsolutions are combined via the median. Since the subproblems can be solved in parallel and since there are fast parallel algorithms for median selection, this technique yields fast parallel algorithms. The important step in this meta-algorithm is median selection. It assures high accuracy.

The research reported in this paper has been supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 378–391, 2017.
https://doi.org/10.1007/978-3-319-72453-9_31

The immediate question, that will be the major focus of this work, is whether the accuracy of a parallelized sequential algorithm matches that of the sequential algorithm (that works on the complete data set). To be me more precise, we consider, say, an SVM algorithm and compare its accuracy on the whole data set to the accuracy of its parallelization via the median trick. Clearly, if the accuracy gets considerable worse by parallelization, the parallelized algorithms are of no practical value. The major insight of our work is a positive answer to the question. Applying the median trick causes only a minor decrease of accuracy.

To answer the question in detail we derive the *parallel ERM principle*, Theorem 1, and conduct computer experiments. ERM is a basic principle in computational and statistical learning theory that can be seen as a reduction from learning to optimization. Theorem 1 can be seen as a parallel or scalable version of this reduction. It states that the generalization error $L_{\mathcal{D}}(h_{med})$ of h_{med}, as computed in Algorithm 1, is with high probability close to the best possible hypothesis, i.e.

$$L_{\mathcal{D}}(h_{med}) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$$

for a hypothesis class $\mathcal{H}$ of bounded VC-dimension and for $\{0, 1\}$-valued loss function ℓ and with generalization error $L_{\mathcal{D}}(h) := \mathbb{E}_{z \sim \mathcal{D}}[\ell(h, z)]$. The benefit of this connection to statistical learning theory is that we immediately get generalization bounds (i.e. bounds on the classification and prediction error of parallel learning via the median trick) and bounds on the sample complexity. The latter relate to the group size in the median trick, Algorithm 1. The VC-dimension and the group size are closely related. In particular, the group size should be roughly a constant multiple of the VC-dimension of the hypothesis class.

We also conduct computer experiments to study the accuracy of implementations of parallel learning on real and artificial data sets. The results are summarized in Table 1 for binary classification on the validation set. We see that the accuracy values of the parallelized algorithms are close to those of the sequential algorithms that work on the whole data set. This is on par with theory. Furthermore, our computer experiments suggest that the median trick also yields meaningful results for problems such as (linear) regression, see Table 2, an insight that does not follow from statistical learning theory and our parallel ERM principle directly. Another reason for our computer experiments is that in practice the assumption of ERM may not be fulfilled. Most critical, the data may not be completely i.i.d. We thus test on real data sets (with the same results as on artificial i.i.d. data). Tables 3 and 4 show the accuracy and fitness on the training set, i.e. the training error.

We describe our method in Sect. 2. We discuss the median and its application (Algorithm 1), and why we can use instead of the median any robust estimator that satisfies some combinatorial property, Definition 1. We will also see that approximate medians can satisfy this property. In Sect. 4, we describe our experiments in detail for binary classification and (linear) regression. We discuss the selected algorithms and data sets. The focus in this section is on the accuracy of learning on (real) data sets. Another important topic that we discuss

in this section is the group size. Our theorems, in particular the parallel ERM principle, can be found in Sect. 3.

1.1 Related Work

Perhaps one of the first applications of the median trick is [12]. Jerrum and Valiant consider the problem of amplifying (or boosting) the success probability of randomized approximate counting algorithms. There have been similar applications to randomized algorithms later on, i.e. the goal is to boost some success probability that is slightly better than $\frac{1}{2}$ to something at least $1 - e^{-r}$ after r repetitions. In contrast, we apply the median trick to learning algorithms (instead of randomized algorithms). Here, the inputs are random. We are primarily interested in generalization bounds (i.e. the quality of learning).

More recently, the median trick or median-of-means has been applied to parameter estimation [15,18,21]. Statistical learning and parameter estimation do not relate easily. Parameter estimation does not yield generalization bounds (in general). Another crucial difference to our work is that [15,21] employ multivariate generalizations of the median, the geometric median and discrete geometric median especially. The univariate median, that we employ, has fast parallel algorithms in comparison to the (discrete) geometric median.

Another approach to parallelization is to solve the underlying optimization problem in parallel or even distributed; see e.g. [4,23] and the references therein. This requires the design, analysis, and implementation of special algorithms for the different (convex) optimization problems. The efficient and parallel running time of our algorithms come from the use of median selection algorithms (or sorting algorithms). Thus there is no need to explicitly design new parallel algorithms given that sequential algorithms are known.

Notable contributions to the design and analysis of algorithms for median selection and sorting are [1–3,5].

2 Method

Our method for classification and regression is a meta-algorithm that applies a given ERM algorithm $\mathcal{A}$ in parallel and combines the results via the median, Algorithm 1. The parallel execution is done by partitioning the input into groups of equal size s, Line 1 in Algorithm 1. Note that we do not use $S_{N+1} = S \setminus (S_1 \cup \cdots \cup S_N)$ in the algorithm. The *group size* s is a parameter of the algorithm that we discuss in detail in Sects. 4 and 3. The ERM algorithm $\mathcal{A}$ solves an (empirical risk) minimization problem, Line 2 in Algorithm 1. This is a restriction of our approach that can be relaxed in practice. We will discuss this issue in Sect. 4.

Empirical risk minimization is the following optimization problem. For a hypothesis class $\mathcal{H}$, a loss function ℓ, and given samples $S := \{(x_1, y_1), \ldots, (x_m, y_m)\}$ that are i.i.d. according to some distribution $\mathcal{D}$ the objective is to find $h \in \mathcal{H}$ that minimizes the empirical risk

$$L_S(h) := \frac{1}{m} \cdot \sum_{i=1}^{m} \ell(h, (x_i, y_i))$$

We denote by $h_S \in \mathcal{H}$ some h that minimizes $L_S(h)$, i.e. some optimal solution to the ERM problem. Examples are given below.

The last step of our meta-algorithm, the median selection, is in Line 3 of Algorithm 1. We define the median in Sect. 2.1. There we also discuss that we can use any univariate robust estimator that satisfies a certain combinatorial property, Definition 1. Most notable, approximate medians, that may be easier to compute, suffice for our need.

Algorithm 1. PARALLELIZE

Input: Samples $S := \{(x_1, y_1), \ldots, (x_m, y_m)\}$
Output: Some hypothesis $h \in \mathcal{H}$
Parameters: Group size s and algorithm $\mathcal{A}$
1. Partition S into $N + 1$ sets $S_1, ..., S_N \subseteq S$ s.t. $|S_i| = s$ for $i = 1, \ldots, N$.
2. For $i = 1, \ldots, N$: Compute

$$h_i := \arg\min_{h \in \mathcal{H}} L_{S_i}(h)$$

and $v_i := L_{S_i}(h_i)$ with $\mathcal{A}$.
3. Compute the median j of $\{v_1, \ldots, v_N\}$ and output $h_{med} := h_j$.

Example 1. Number of misclassifications. We define the loss $\ell_\#(h, (x, y))$ as 1 if $h(x) = y$ and as 0 otherwise for $x \in \mathbb{R}^d$ and $y \in \{-1, +1\}$. Other domains than $\mathbb{R}^d$ are possible. The resulting ERM is NP-hard if, for example, the hypothesis class $\mathcal{H}$ is the class of all linear functions.

Example 2. Hinge loss. This is a standard loss function in binary classification and due to [10]. We define it as $\ell_{hinge}(h, (x, y)) := \max(0, 1 - y \cdot h(x))$ for $x \in \mathbb{R}^d$, $y \in \{-1, +1\}$, and $h : \mathbb{R}^d \to \{-1, +1\}$. The optimization problem, i.e. finding the minimum to $L_S(h)$, is convex for linear h. This case is also known as linear Support Vector Machines (SVM), see e.g. [6]. To apply Algorithm 1, we set the group size to a constant multiple of the VC-dimension of $\mathcal{H}$, which is d in our case. To solve the hinge loss optimization problem in Line 2, we can use any linear SVM algorithm.

Example 3. Least squares loss. This is a standard loss function in regression: $\ell_2(h, (x, y)) := (y - h(x))^2$. Similar to hinge loss optimization, the resulting ERM rule $L_S(h)$ is convex for the class of linear h and also known as Ordinary Least Squares. Here, $x \in \mathbb{R}^d$, $y \in \mathbb{R}$, and $h : \mathbb{R}^d \to \mathbb{R}$. A rule of thumb, that we study in detail in Sect. 4, is to set the group size to a constant multiple of the dimension d.

2.1 Medians

Our definition of the median slightly deviates from more common definitions. For a set $P \subseteq \mathbb{R}$ and $x \in P$, let $rank_P(x)$ be the position of x in the sorted sequence of numbers in P. Let $n := |P|$. The *median* of P is $x \in P$ with $rank_P(x) = \lfloor \frac{n}{2} \rfloor$.

We denote it by $med(P)$. (We could also have chosen $rank_P(x) = \lceil \frac{n}{2} \rceil$.) An $x \in P$ is a *β-approximate median* with rank $\lfloor \frac{n+1}{2} \rfloor - \frac{\beta n}{2} \leq rank_P(x) \leq \lceil \frac{n+1}{2} \rceil + \frac{\beta n}{2}$. A *$\beta$-approximate median estimator*, $0 \leq \beta \leq 1$, is any function f, defined over any finite set of reals, such that $f(P)$ is a β-approximate median for any finite $P \subseteq \mathbb{R}$.

The following definition of a robust estimator captures the combinatorial nature of the median and other estimators, see Lemma 1. It is this property that we need for the proof of the parallel ERM, Theorem 1.

Definition 1. *Let $0 \leq \alpha \leq 1$ and T be a real-valued function defined over any finite set of reals. Then, T is an* α-robust estimator *iff for all finite $P \subseteq \mathbb{R}$ and for all closed intervals $I \subseteq \mathbb{R}$ the following property holds:*

$$|P \cap I| > \alpha |P| + 1 \text{ implies } T(P) \in P \cap I.$$

Lemma 1. *Any β-approximate median estimator is a $(\frac{1+\beta}{2})$-robust estimator. In particular, the median is a $\frac{1}{2}$-robust estimator.*

Proof. Let T be a β-approximate median estimator. Fix some $P \subseteq \mathbb{R}$ with $|P| = n$. Assume that $|P \cap I| > (\frac{1+\beta}{2})n + 1$. Let $I = [a, b]$. Let L be the left half of P, i.e. $L := \left\{ x \in P : rank_P(x) < \lfloor \frac{n+1}{2} \rfloor - \frac{\beta n}{2} \right\}$, and R be the right part of P, i.e. $R := \left\{ x \in P : rank_P(x) > \lceil \frac{n+1}{2} \rceil + \frac{\beta n}{2} \right\}$, and let $M := P \setminus (L \cup R)$ be the middle part. Define the intervals $L' := (-\infty, \max(L)]$ and $R' := [\min(R), +\infty)$. If $a \in L'$ and $b \in R'$, then $T(P) \in P \cap I$ since I is an interval. If $a \notin L'$ or $b \notin R'$, then $P \cap I$ contains elements either only from $L \cup M$ or only from $R \cup M$. It holds that $|L| = |R| = \lfloor \frac{n+1}{2} \rfloor - 1 - \left\lfloor \frac{\beta n}{2} \right\rfloor$. Thus $|L \cup M| = n - |R| \leq (\frac{1}{2} + \beta)n + 1$ and $|R \cup M| = n - |L| \leq (\frac{1}{2} + \beta)n$. This implies $|P \cap I| \leq |(L \cup M) \cap I| \leq (\frac{1}{2} + \beta)n + 1$ or $|P \cap I| \leq |(R \cup M) \cap I| \leq (\frac{1}{2} + \beta)n + 1$. A contradiction in both cases to our assumption that $|I \cap P| > (\frac{1+\beta}{2})n + 1$. To see that $n - |R| \leq (\frac{1}{2} + \beta)n + 1$ and $n - |L| \leq (\frac{1}{2} + \beta)n + 1$, note that $n - \lfloor \frac{n+1}{2} \rfloor \leq \frac{n}{2}$ and $\left\lfloor \frac{\beta n}{2} \right\rfloor \leq \frac{\beta n}{2}$. The second claim of the lemma follows from the first by setting $\beta = 0$. □

2.2 Discussion of Method

Remark 1. Robustness. The median is robust against outliers in data. See for example [26]. This robustness, formulated as a high breakdown in robust statistics [26], carries over to Algorithm 1: If less than half of the groups contain one or more outliers, Algorithm 1 still yields proper models. This is also true for our notion of α-robust estimators, Definition 1. It is easy to see that an α-robust estimator exhibits a high breakdown point.

Remark 2. Computation. Median selection and rank computations can be done in deterministic sequential time $O(n)$ [3]. In practice, if no such implementation is available, we can use sorting. This yields algorithms of running time $O(n \log(n))$. We can use parallel sorting algorithms [1] that employ up to $p \leq n$ processors

in parallel to find the median in deterministic time $O(\frac{n}{p}\log(n))$. The sorting networks (algorithms) in [1] are however not used in practice due to their (combinatorial) complexity. Implementations usually run in time $O(\frac{n}{p}\log(n)^2)$ [2]. Moreover, this computational approach fits into distributed computing environments such as MapReduce [8] and more generally in models similar to Valiant's model of bulk-synchronous parallel computation [31]. Regarding the fastest parallel algorithms with running time bounds, the algorithm of Cole and Yap [5] runs in $O((\log\log n)^2)$ parallel deterministic time on $O(n)$ processors in Valiant's model.

Remark 3. Online learning. Although we are not dealing with it here, a possible approach to online learning is simply to adapt our meta-algorithm for it. Assume that the samples are presented one by one. We form the groups by adding new elements to the group S_N as long as $|S_N| < s$. If a $|S_N| = s$, we solve the ERM on S_N and update the median of $\{v_1, \ldots, v_N\}$ (and continue with a new group S_{N+1}). This can be done in time $O(\log(N))$ and with storage $O(N)$.

Remark 4. Alternatives to the median. Our definition of robust estimators, Definition 1, is related to the breakdown point of estimators of location. High breakdown estimators may be an alternative to the median. For example, Rousseeuw and Leroy [26] (p. 158) remark that univariate, high breakdown estimators arise from robust linear regression. Another option may be univariate mode estimators with a high breakdown point. However, to the best of our knowledge, none of these estimators can be computed faster than the median. The most interesting alternatives to the computation of the medians seem to be algorithms for approximate medians, as defined above.

3 Theorems

We prove our main theorem, Theorem 1. We recall that the *VC-dimension* (see Definition 6.5 in [27] on p. 70) of a hypothesis class $\mathcal{H}$ of functions h of the form $\mathcal{X} \to \{0,1\}$ is the maximal size of a set $C \subset \mathcal{X}$ that can be shattered by $\mathcal{H}$, where a set $\mathcal{H}$ *shatters* C if the restriction of $\mathcal{H}$ to C is the set of all functions from C to $\{0,1\}$. We denote it by $VCdim(\mathcal{H})$. If $\mathcal{H}$ can shatter sets of arbitrarily large size we say that $\mathcal{H}$ has infinite VC-dimension. As an example, $VCdim(\mathcal{H}) \leq \log(|\mathcal{H}|)$ for every finite $\mathcal{H}$.

Theorem 1. *Parallel Empirical Risk Minimization. Let $\mathcal{H}$ be a hypothesis class of VC-dimension d and let ℓ be a $\{0,1\}$-valued loss function. Let $s \in \mathbb{N}$, S be a set of i.i.d. samples, and $S_1, \ldots, S_{N+1}$ be a partition of S such that $|S_i| = s$ for all $i \in \{1, \ldots, N\}$. Let $\epsilon > 0$. Let h_{med} be as in Algorithm 1. If*

$$s \geq C_2 \cdot 16 \cdot \frac{d + \log(15)}{\epsilon^2}$$

then the probability over S that

$$L_{\mathcal{D}}(h_{med}) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$$

holds is at least $1 - 4 \cdot e^{-N}$. *(The constant* C_2 *is independent of* $\mathcal{H}, \ell, \epsilon$. *See Theorem 2 below.)*

Before we continue with proof of the theorem, we discuss shortly the necessary tools from statistical learning theory. A set S of samples that are i.i.d. according to some $\mathcal{D}$ is ϵ-*representative*, $\epsilon > 0$, iff for all $h \in \mathcal{H}$, $|L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon$. We say that a hypothesis class $\mathcal{H}$ has the *uniform convergence property* if there exists a function $m_{\mathcal{H}}^{UC} : (0,1)^2 \to \mathbb{N}$ such that for every $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over Z, if S is a sample of $m \geq m_{\mathcal{H}}^{UC}(\epsilon, \delta)$ examples drawn i.i.d. according to $\mathcal{D}$, then, with probability of at least $1 - \delta$, S is ϵ-representative.

The uniform convergence property is essentially equivalent with the prerequisites of the theorem. One direction is stated in the following theorem.

Theorem 2 *(Theorem 6.8 in [27] on p. 72). Let* $\mathcal{H}$ *be a hypothesis class of functions from a domain* $\mathcal{X}$ *to* $\{0,1\}$ *and let the loss function be* $\{0,1\}$*-valued. Assume that* $VCdim(H) = d < \infty$. *Then, there are absolute constants* C_1, C_2 *such that* $\mathcal{H}$ *has the uniform convergence property with sample complexity* $C_1 \cdot \frac{d+\log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq C_2 \cdot \frac{d+\log(1/\delta)}{\epsilon^2}$.

Our proof relies on the uniform convergence property. Together with the median trick, an application of a Chernoff bound, and the previous theorem, we can prove our theorem.

Proof. Let S_i be the i-th group and h_i be the ERM rule minimizer and let h^* minimize $L_{\mathcal{D}}(\cdot)$. Assume that each group S_i is ϵ'-representative with probability of at least $1 - p_0$ (we will fix it below) and with $\epsilon' := \frac{\epsilon}{4}$. Then, for all $i \in \{1, \ldots, N\}$:

$$|L_{S_i}(h_i) - L_{\mathcal{D}}(h_i)| \leq \epsilon'$$

by definition (and with probability of at least $1 - p_0$) and

$$|L_{\mathcal{D}}(h_i) - L_{\mathcal{D}}(h^*)| \leq 2\epsilon'$$

since $L_{\mathcal{D}}(h_i) \leq L_{\mathcal{D}}(h^*) + 2\epsilon'$. This implies

$$|L_{S_i}(h_i) - L_{\mathcal{D}}(h^*)| \leq 3\epsilon' \tag{1}$$

also with probability of at least $1 - p_0$. Then, if with probability of at least $1 - p_1$,

$$|L_S(h_{med}) - L_{\mathcal{D}}(h^*)| \leq 3\epsilon'$$

holds, and if with probability of at least $1 - p_2$,

$$|L_S(h_{med}) - L_{\mathcal{D}}(h_{med})| \leq \epsilon'$$

holds, then

$$|L_{\mathcal{D}}(h_{med}) - L_{\mathcal{D}}(h^*)| \leq 4\epsilon' = \epsilon$$

holds with probability of at least $1 - p_1 - p_2$ by the union bound.

It remains to give upper bounds on the probabilities p_1 and p_2 and to fix p_0. For p_1 we apply the median trick. We apply Theorem 2 for both bounds. We start with p_1 and the median trick. Let W_i be 1 if Eq. 1 does *not* hold and 0 otherwise. Define $W := \sum_{i=1}^{N} W_i$. Then,

$$\Pr(|L_S(h_{med}) - L_{\mathcal{D}}(h^*)| > 3\epsilon') \leq \Pr(W > \frac{N}{2} + 1) \tag{2}$$

by Lemma 1. To see this, observe that $|L_S(h_{med}) - L_{\mathcal{D}}(h^*)| > 3\epsilon'$ is equivalent to $L_S(h_{med}) \notin I := [L_{\mathcal{D}}(h^*) - 3\epsilon', L_{\mathcal{D}}(h^*) + 3\epsilon']$ which is equivalent to $med(P) \notin P \cap I$ for $P := \{v_1, \ldots, v_N\}$. By Lemma 1 and by the definition of robust estimators, Definition 1, we conclude that $|P \cap I| \leq \frac{1}{2} \cdot |P| + 1 = \frac{N}{2} + 1$ which is equivalent to $|P \cap \bar{I}| > \frac{N}{2} + 1$. This proves Eq. 2.

We want to apply a Chernoff bound. Let $\mu := \mathbb{E}(W)$. Then, for any $c > 1$, $\Pr(W \geq (1+c)\mu) \leq e^{-(c/3)\mu}$. Note that the random variables W_i are mutually independent since the samples in S are i.i.d. and that $\mu \leq p_0 N$. Also note that if $(1+c)\mu \leq (1+c)p_0 N < N/2 + 1$, then $\Pr(W > \frac{N}{2} + 1) \leq \Pr(W \geq (1+c)\mu)$. Fix $c := 6$. We thus have to ensure that $7\mu \leq 7p_0 N < N/2 + 1$ by applying Theorem 2 appropriately. We set $\epsilon = 4\epsilon'$ and $p_0 = \delta = 1/15$. We have: $\frac{7}{15} < \frac{1}{2} + \frac{1}{N}$ for all $N \geq 1$. Thus, we have to assume that the group size s is at least $C_2 \cdot 16 \cdot \frac{d + \log(15)}{\epsilon^2} \geq m_{\mathcal{H}}^{UC}(4\epsilon', \frac{1}{15})$ to get:

$$p_1 \leq \Pr(W > (1+c)\mu) \leq e^{-(c/3)N/2} = e^{-N}$$

To derive a bound for p_2, we directly apply Theorem 2 for all the $m' = N \cdot s$ samples in the partition $S_1, \ldots, S_N$ and with $\epsilon = 4\epsilon'$ and with δ to be determined next. We have $N \cdot C_2 \cdot 16 \cdot \frac{d + \log(15)}{\epsilon^2} \leq N \cdot s = m'$ and $m_{\mathcal{H}}^{UC}(4\epsilon', \delta) \leq m' \leq C_2 \cdot 16 \cdot \frac{d + \log(1/\delta)}{\epsilon^2}$ by Theorem 2. Putting it together:

$$N \cdot C_2 \cdot 16 \cdot \frac{d + \log(15)}{\epsilon^2} \leq m' \leq C_2 \cdot 16 \cdot \frac{d + \log(1/\delta)}{\epsilon^2}$$

iff

$$N \cdot (d + \log(15)) - d \leq \log(1/\delta)$$

which implies

$$p_2 \leq e^{-(N \cdot (d + \log(15)) - d)} = e^{-(N-1) \cdot d}$$

Thus we have $1 - p_1 - p_2 \geq 1 - e^{-N} - e^{-(N-1)d} \geq 1 - 4 \cdot e^{-N}$. □

4 Experiments

Here, we study the accuracy of Algorithm 1 and its single parameter, the group size s, and compare the accuracy of $\mathcal{A}$ and the accuracy of its parallelization. The results for binary classification are in Tables 1 and 3. These results are on par with theory, Theorem 1, and show that the accuracy of Algorithm 1 drops

slightly in comparison to $\mathcal{A}$ (when run on the complete data set). We discuss these results together with similar results on (linear) regression, Tables 2 and 4, in detail in Sect. 4.1. In Sect. 4.2, we derive a heuristic for how to set the group size s. Finally, we discuss in Sect. 4.3 why a slight but very natural variation of Algorithm 1 does not yield accurate results.

Table 1. Results of classification experiments, validation (or test) set; higher numbers are better

$s = 20d$	C-SVM		Log. Regr.		LDA		AdaBoost		KNN	
	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2
Art. data	91.40	83.40	91.40	89.35	90.65	90.80	90.95	87.45	90.85	90.70
Haberm. s. [17]	74.19	70.97	74.19	72.58	74.19	74.19	61.29	74.19	77.42	62.90
Blood [33]	78.00	78.67	79.33	78.00	78.67	78.00	74.67	76.00	76.67	69.33
Prima [17]	79.87	79.87	79.22	77.92	79.22	77.92	77.27	78.57	78.57	68.83
Tic tac toe [17]	62.50	62.50	64.58	60.42	64.06	58.33	100.00	94.27	81.25	72.40
Mammogr. [9]	83.13	81.93	84.34	84.34	81.33	80.12	75.90	81.93	80.72	80.12
Monks 3 [17]	83.78	86.49	78.38	83.78	81.98	83.78	98.20	99.10	99.10	93.69
Mushroom [17]	98.67	96.46	95.04	99.56	93.98	94.42	99.73	99.73	100.00	96.37
Magic [17]	79.71	79.21	80.02	78.42	79.15	77.00	88.91	82.47	81.23	73.13
Bank [22]	87.87	88.84	89.14	88.40	89.19	88.50	90.39	88.54	87.92	87.87
Adult [17]	80.72	79.93	81.91	79.06	81.08	78.60	84.25	81.11	77.01	72.69

4.1 Comparison of Accuracy

In this chapter we present experimental results for empirical risk minimization problems on artificial and real data (binary classification and linear regression). All except one of the real world data sets are taken from the UCI machine learning repository [17]. The additional data set consists of energy data of a photovoltaic system for several months. Prior to the experiments the data sets are cleaned up by removing rows containing missing values. For each method we set the hyper-parameters to default values.

In order to get unbiased evaluation results, we split each data set in a training and test set (80 : 20). On the one hand, we train models via conventional empirical risk minimization on the whole training set. On the other hand, we apply our parallel empirical risk minimization approach on the training set. The goal is to test if the two resulting models perform equally well (i.e. result in similar accuracy values) on the test set (ACC 1 is based on conventional ERM, ACC 2 is based on parallel ERM). Hence, the crucial aspect is the deviation between both accuracy values. In the classification experiments accuracy indicates the proportion of correctly classified samples and in the regression experiments the accuracy values correspond to the root-mean-square error calculated by $\sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$.

In Tables 1 and 2 we list results of these classification and regression experiments on the validation set. Each row represents the results for one particular

Table 2. Results of regression experiments, validation (or test) set; lower numbers are better

$s = 30d$	eps-SVM		L1		L2		LTS		KNN	
	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2
Art. data	34.80	33.97	34.93	35.23	31.28	31.84	37.26	37.17	34.07	32.77
Compr. str. [32]	11.53	11.30	11.63	12.12	10.91	11.07	16.78	15.99	9.64	11.84
Skillcraft1 [28]	0.97	0.98	0.97	0.98	0.97	0.98	0.98	0.98	1.07	1.10
Wine qual. [7]	0.75	0.78	0.75	0.76	0.75	0.77	0.75	0.76	0.82	0.87
Parkinsons [29]	7.63	7.91	7.64	8.05	7.52	7.86	7.88	8.03	5.77	8.19
Power pl. [13,30]	4.57	4.63	4.57	4.78	4.56	4.67	4.57	4.69	3.91	6.16
Year pred. [17]	9.79	9.83	9.80	9.94	9.70	9.88	10.04	10.25	11.44	11.41
Photovoltaic	80.20	82.55	82.53	79.93	79.10	79.43	165.55	150.82	42.56	71.01

Table 3. Results of classification experiments, training set; higher numbers are better

$s = 20d$	C-SVM		Log. Regr.		LDA		AdaBoost		KNN	
	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2
Art. data	90.73	83.19	90.74	88.16	90.81	90.78	100.00	87.51	92.94	90.50
Haberm. s. [17]	75.41	74.18	75.82	76.23	75.41	74.18	98.36	77.87	78.28	77.05
Blood [33]	75.75	76.25	76.59	74.75	76.42	76.25	93.48	74.08	81.44	70.23
Prima [17]	76.38	75.90	77.36	77.52	77.20	76.38	100.00	82.08	79.32	73.29
Tic tac toe [17]	66.06	66.06	71.02	65.40	70.76	67.62	100.00	95.43	86.03	74.02
Mammogr. [9]	82.98	80.72	83.13	84.04	81.02	82.08	93.52	83.58	84.94	78.61
Monks 3 [17]	79.23	80.14	76.98	79.23	77.65	81.04	99.10	98.87	98.87	95.94
Mushroom [17]	98.74	95.22	94.71	99.76	93.13	93.62	99.89	99.53	99.98	96.52
Magic [17]	79.15	78.44	78.85	78.27	78.30	77.55	100.00	81.89	85.88	71.88
Bank [22]	88.41	88.89	89.09	88.37	89.22	88.70	100.00	88.41	91.10	88.17
Adult [17]	80.87	80.31	82.06	79.09	81.11	78.62	99.99	80.95	83.03	72.33

data set. The group size is set to $s = 20d$ (classification) and $s = 30d$ (regression). In addition, we also provide the training accuracy values in Tables 3 and 4.

We can conclude that the parallel ERM approach results in very adequate models compared to standard ERM both on artificial and on real world data. Due to the parallelization of ERM the running time can be reduced significantly. Concerning our classification experiments the running times can be reduced on average by 34.34% (C-SVM), 9.76% (Log. Regr.) and 54.36% (AdaBoost). Especially for high-dimensional data a large reduction of running time is possible. Concerning our regression experiments the running times can be reduced on average by 57.58% (L1), 61.57% (L2), 54.96% (LTS) and 70.53% (eps-SVM). All experiments are performed using the software environment R [24] and appropriate R packages [11,14,16,19,20,25]. In general, we expect that the running time of our parallelization scales well with the number of processors (or computing cores).

Table 4. Results of regression experiments, training set (i.e. model fitness); higher numbers are better

$s = 30d$	eps-SVM		L1		L2		LTS		KNN	
	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2	ACC 1	ACC 2
Art. data	33.71	33.01	33.84	34.11	30.74	31.16	36.06	35.97	27.19	32.62
Compr. str. [32]	10.54	10.39	10.58	11.04	10.22	10.50	14.44	13.81	7.29	10.31
Skillcraft1 [28]	0.99	1.01	0.99	7.38	0.99	10.96	12.50	10.15	0.87	1.04
Wine qual. [7]	0.75	0.78	0.75	0.76	0.75	0.76	0.76	0.77	0.66	0.86
Parkinsons [29]	7.62	7.81	7.64	8.06	7.46	7.71	7.76	8.09	4.48	7.82
Power pl. [13,30]	4.57	4.68	4.57	4.81	4.56	4.71	4.58	4.73	3.22	6.11
Year pred. [17]	9.64	9.78	9.67	9.77	9.35	9.45	9.94	10.10	9.15	10.65
Photovoltaic	79.92	82.33	82.74	80.67	78.72	78.87	168.46	153.18	32.15	69.79

4.2 Group Size

How to set the group size s? Hyper-parameter optimization is particular simple here since s is monotone, i.e. the larger s the better the accuracy. Thus, a binary search that looks for the smallest s that satisfies accuracy or running time requirements will work.

A different, heuristic approach that avoids hyper-parameter search is to set the group size between $10d$ to $30d$ where d is the "dimension". In most situations, d is the dimension of the point cloud or the number of features. The motivation comes from the proof of Theorem 1. To apply the median trick the event $L_{\mathcal{D}}(h_i) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$ should hold with some probability bounded away from $\frac{1}{2}$, say at least $\frac{2}{3}$. (The event states that the error $L_{\mathcal{D}}(h_i)$ of h_i as in Algorithm 1 is not much worse than the error $\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$ best possible for the hypothesis class $\mathcal{H}$.) This motivates the rule of thumb in situations in which a linear dependency on d suffices. Note however that learning or estimating more complex models than we did in our experiments, e.g. covariance matrices, may require other group sizes. For example $s = \Omega(d \log(d))$ or $s = \Omega(d^2)$ for covariance matrices. In case of learning the probability table of d binary variables we may need at least $s = \Omega(2^d)$ samples per group.

4.3 Min-of-Min Counterexample

Another natural variant of our parallel ERM technique would be to apply the minimum instead of the median. One advantage of the minimum is the computational efficiency. Computing the minimum can be done slightly faster. Both problems can be solved in linear time. Nevertheless, this approach leads to inadequate models under certain circumstances. In order to illustrate this behavior we train a binary C-SVM classifier on a two-dimensional artificial data set and apply conventional ERM, parallel ERM via the median trick and parallel ERM using the minimum. The results are visualized in Fig. 1. Whereas the application of the median results in an appropriate model, the minimum is not a good choice in this case.

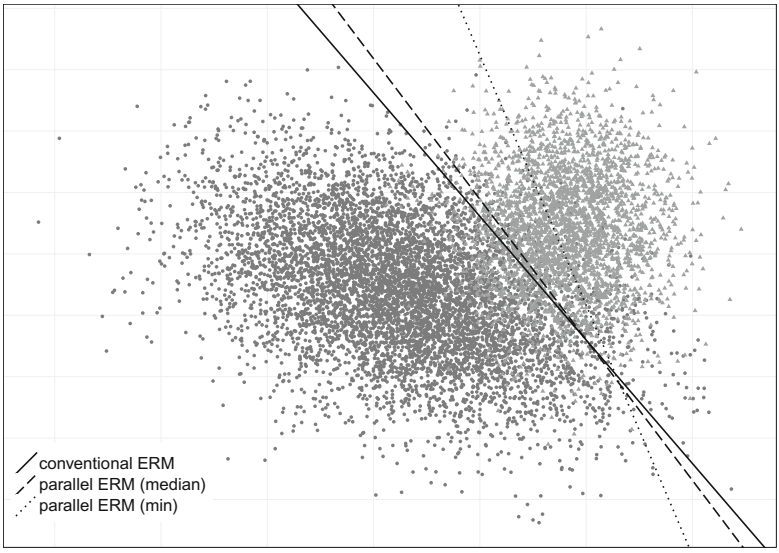


Fig. 1. Conventional ERM, parallel ERM (median) and parallel ERM (min)

5 Summary

We showed that the median trick applies to ERM and related problems. It yields fast parallel, scalable algorithms for important learning problems. The connection to statistical learning theory has helped us to identify and solve some problems. Most notable, we derived bounds and heuristics for the adequate group size. The median trick is an appropriate technique if we want to use multiple processors (or computing cores) to speed-up machine learning, especially in case of mid-sized to large data sets for which the known sequential algorithms are too slow.

References

1. Ajtai, M., Komlós, J., Szemerédi, E.: An O(n log n) sorting network. In: Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, pp. 1–9 (1983)
2. Batcher, K.E.: Sorting networks and their applications. In: Proceedings of the AFIPS Spring Joint Computer Conference, pp. 307–314 (1968)
3. Blum, M., Floyd, R.W., Pratt, V., Rivest, R.L., Tarjan, R.E.: Linear time bounds for median computations. In: Proceedings of the 4th Annual ACM Symposium on Theory of Computing, pp. 119–124 (1972)
4. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011)
5. Cole, R., Yap, C.: A parallel median algorithm. Inf. Process. Lett. **20**(3), 137–139 (1985)

6. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**, 273–297 (1995)
7. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J.: Modeling wine preferences by data mining from physicochemical properties. Decis. Support Syst. **47**, 547–553 (2009)
8. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. In: Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation, vol. 6, p. 10 (2004)
9. Elter, M., Schulz-Wendtland, R., Wittenberg, T.: The prediction of breast cancer biopsy outcomes using two cad approaches that both emphasize an intelligible decision process. Med. Phys. **34**(11), 4164–4172 (2007)
10. Gentile, C., Warmuth, M.K.: Linear hinge loss and average margin. In: Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II, pp. 225–231 (1999)
11. Harrell, F.E.J., Dupont, C.: Hmisc: Harrell Miscellaneous. R package version 4.0-2 (2016). https://CRAN.R-project.org/package=Hmisc
12. Jerrum, M.R., Valiant, L.G., Vazirani, V.V.: Random generation of combinatorial structures from a uniform distribution. Theor. Comput. Sci. **43**, 169–188 (1986)
13. Kaya, H., Tüfekci, P.: Local and global learning methods for predicting power of a combined gas & steam turbine. In: Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering ICETCEE 2012, pp. 13–18 (2012)
14. Koenker, R.: Quantreg: Quantile Regression. R package version 5.29 (2016). https://CRAN.R-project.org/package=quantreg
15. Kogler, A., Traxler, P.: Efficient and robust median-of-means algorithms for location and regression. In: Proceedings of the 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2016), pp. 206–213 (2016)
16. Kuhn, M.: Caret: Classification and Regression Training. R package version 6.0-73 (2016). https://CRAN.R-project.org/package=caret
17. Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml
18. Lugosi, G., Mendelson, S.: Risk minimization by median-of-means tournaments. In: Presented at Foundations of Computational Mathematics (FoCM 2017), Barcelona (2017)
19. Mersmann, O.: Microbenchmark: Accurate Timing Functions. R package version 1.4-2.1 (2015). https://CRAN.R-project.org/package=microbenchmark
20. Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F.: e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.6-8 (2017). https://CRAN.R-project.org/package=e1071
21. Minsker, S.: Geometric median and robust estimation in Banach spaces. Bernoulli **21**(4), 2308–2335 (2015)
22. Moro, S., Cortez, P., Rita, P.: A data-driven approach to predict the success of bank telemarketing. Decis. Support Syst. **62**, 22–31 (2014)
23. Peteiro-Barral, D., Guijarro-Berdiñas, B.: A survey of methods for distributed machine learning. Prog. Artif. Intell. **2**(1), 1–11 (2013)
24. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2017). https://www.R-project.org/

25. Rousseeuw, P., Croux, C., Todorov, V., Ruckstuhl, A., Salibian-Barrera, M., Verbeke, T., Koller, M., Maechler, M.: Robustbase: Basic Robust Statistics. R package version 0.92-7 (2016). http://CRAN.R-project.org/package=robustbase
26. Rousseeuw, P.J., Leroy, A.M.: Robust Regression and Outlier Detection. Wiley, Hoboken (2005)
27. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning - From Theory to Algorithms. Cambridge University Press, Cambridge (2014)
28. Thompson, J.J., Blair, M.R., Chen, L., Henrey, A.J.: Video game telemetry as a critical tool in the study of complex skill learning. PLoS ONE **8**(9), 1–12 (2013)
29. Tsanas, A., Little, M.A., McSharry, P.E., Ramig, L.O.: Accurate telemonitoring of Parkinson's disease progression by noninvasive speech tests. IEEE Trans. Biomed. Eng. **57**, 884–893 (2010)
30. Tüfekci, P.: Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. Int. J. Electr. Power Energy Syst. **60**, 126–140 (2014)
31. Valiant, L.G.: A bridging model for parallel computation. Commun. ACM **33**(8), 103–111 (1990)
32. Yeh, I.C.: Modeling of strength of high-performance concrete using artificial neural networks. Cem. Concr. Res. **28**(12), 1797–1808 (1998)
33. Yeh, I.C., Yang, K.J., Ting, T.M.: Knowledge discovery on RFM model using Bernoulli sequence. Expert Syst. Appl. **36**(3), 5866–5871 (2009)

Mathematical Aspects of Information Security and Cryptography

Leakage-Resilient Riffle Shuffle

Paweł Lorek[2], Michał Kulis[1], and Filip Zagórski[1(✉)]

[1] Department of Computer Science, Faculty of Fundamental Problems of Technology, Wrocław University of Science and Technology, Wrocław, Poland
filip.zagorski@pwr.edu.pl

[2] Faculty of Mathematics and Computer Science, Mathematical Institute, Wrocław University, Wrocław, Poland

Abstract. Analysis of various card-shuffles – finding its mixing-time is an old mathematical problem. The results show that *e.g.,* it takes $\mathcal{O}(\log n)$ riffle-shuffles (Aldous and Diaconis, *American Mathematical Monthly*, 1986) to shuffle a deck of n cards while one needs to perform $\Theta(n \log n)$ steps via cyclic to random shuffle (Mossel et al., *FOCS*, 2004).

Algorithms for generating pseudo-random permutations play a major role in cryptography. *Oblivious* card shuffles can be seen as block ciphers (and *e.g.,* may be used for format-preserving encryption) while *non-oblivious* card shuffles often are a building block for cryptographic primitives (*e.g.,* Spritz, RC4).

Unfortunately, all results about the mixing times of card shuffling algorithms are in the black-box model. The model does not capture real-world capabilities of adversaries who may be able to *e.g.,* obtain some information about the randomness used during the shuffling. In this paper we investigate the impact on the mixing time of the riffle shuffle by an adversary who is able to eavesdrop some portion of the random bits used by the process. More precisely: assuming that each bit of the randomness leaks independently with probability p we show that whenever *RiffleSST* performs $r = \log_{\frac{2}{2-(1-p)^2}} \binom{n}{2} + \log_{\frac{2}{2-(1-p)^2}} \left(\frac{1}{\varepsilon n!}\right)$ steps, it cannot be distinguished from a permutation selected uniformly at random with the advantage larger than ε.

Keywords: Leakage resilience
Pseudo-random permutation generator · Markov chains · Mixing time
Card shuffle · Riffle Shuffle · Stream cipher · Distinguisher

1 Introduction

1.1 Card Shuffling and Cryptography

Shuffling procedures (or card shuffles) are used as cryptographic building blocks. A card shuffle as a way to obtain a permutation can be seen as a block cipher.

Authors were supported by Polish National Science Centre contract number DEC-2013/10/E/ST1/00359.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 395–408, 2017.
https://doi.org/10.1007/978-3-319-72453-9_32

Due to efficiency reasons, only *oblivious* card shuffles are good candidates for block ciphers (*e.g.,* [14]). *Non-oblivious* card shuffles need to be used differently (*e.g.,* as a key scheduling algorithm [13]). But card shuffles may also help to design/describe higher-level systems *e.g.,* ones which goal is to achieve anonymity like: Private Information Retrieval schemes [10,22] or mixing with application to voting [7,9].

Oblivious Shuffles. The applicability of *oblivious* card shuffles to cryptography was noticed many years ago by *e.g.,* Naor and Reingold [16] for Thorp shuffle. Oblivious shuffles can be seen as block ciphers: suppose that a deck has $n = 2^l$ cards then the message space and the ciphertext space is equal to the set of all binary strings of length l. Randomness used by the process corresponds to the trajectory of a given card, and one does not need to trace trajectories of other cards. This point of view led to the proposals [8,14,15,17] (useful for *e.g.,* format preserving encryption schemes) with provable properties in the black-box model (meaning that an adversary has only access to inputs and output of the shuffling algorithm like in CPA-experiment [chosen-plaintext attack]).

Non-oblivious Shuffles. In *non-oblivious* shuffles one needs to trace a trajectory of every of the n cards to be able to tell what is the final position of a single card. Because of that non-oblivious shuffles are used as building blocks of cryptographic schemes (in *e.g.,* [12], and especially as Key Scheduling Algorithms in *e.g.,* RC4, Spritz [18], *etc.*) rather than being used as encryption schemes.

Then the security of a cryptographic scheme which uses some card shuffling as a building block depends on the quality of the shuffle. This can be measured by how a given shuffling is close to the uniform distribution (over all possible permutations). This depends on:

1. the rate of convergence to the stationary distribution (depends on the shuffling algorithm itself);
2. the number of steps made by the algorithm. (In particular we are interested in the number of steps needed so that the distribution of the chain at the given step is close to uniform one.)

One of the weaknesses found in RC4 is that its Key Scheduling Algorithm (KSA) makes only n steps while the rate of convergence is $O(n \log n)$ [11,13].

1.2 Leakage

Classically, in (black-box) cryptography, the security definitions assume that an attacker can have only access to inputs and outputs of a cryptographic scheme – for instance, for encryption one considers CPA-security (Chosen Plaintext Attack) or CCA-security (Chosen Cipertext Attack). These definitions assume that no information about the secret-key (or some internal computations) is leaked. In reality however, a device (or particular scheme or protocol implementation) may expose to an adversary lots of additional information, an adversary may measure all kinds of side-channels *e.g.,* electromagnetic [6], acoustic [5] *etc.* One of the most powerful kind of side-channel attacks are *timing*

attacks (because an adversary may perform them remotely), see [1,21]; or the combination of techniques [23].

Practice shows that an adversary may obtain some direct information about the secret key: assume that $\mathbf{b} = (b_1, \ldots, b_t)$ bits of key (or of a function of key) are used in a given round of the algorithm. Then the Hamming weight $HW(\mathbf{b}) = \sum_{i=1}^{t} b_i$ can leak. More precisely Hamming weight with some Gaussian noise leaking was considered *e.g.,* in [19–21].

1.3 Our Contribution

In this paper we consider the model where each bit b_i of the randomness used by the shuffling algorithm leaks independently with some prescribed probability $p \in [0, 1)$. We analyze a single run of the Riffle Shuffle and our goal is to find the number of rounds the algorithm needs to make in order not to reveal *any* information about the resulting permutation (in the presence of an eavesdropping adversary).

We analyze a non-oblivious shuffling algorithm called *RiffleSST* that is leakage resilient. We show that even if an adversary $\mathcal{A}$ learns each bit of the key K with probability p (knowledge of bits of the key are denoted by $\Lambda_p(K)$) it cannot tell much about the resulting permutation. Putting this in other words: even if an adversary knows some bits of the key $\Lambda_p(K)$, it cannot distinguish the permutation produced by r rounds of the *RiffleSST* algorithm from the permutation sampled from the uniform distribution with probability better than ε.

The contribution of this paper is the first analysis of a card shuffle algorithm in the presence of a randomness-eavesdropper. The result is formulated as Theorem 1.

Theorem 1. *Let $\mathcal{A}$ be an adversary. Let $K \in \{0,1\}^{rn}$ be a secret key. Let $\Lambda_p(K)$ be the random variable representing the leakage of the key such that $\mathcal{A}$ learns each bit of the key independently at random with probability p. Let $\mathsf{S}_{r,n}(K)$ be RiffleSST shuffle of n cards which runs for*

$$r = \log_{\frac{2}{2-(1-p)^2}} \binom{n}{2} + \log_{\frac{2}{2-(1-p)^2}} \left(\frac{1}{\varepsilon n!} \right)$$

steps with $0 < \varepsilon < 1/n!$, then

$$\left| \Pr_{K \leftarrow \{0,1\}^{rn}} [\mathcal{A}(\Lambda_{p,r}, \mathsf{S}_{r,n}(K)) = 1] - \Pr_{R \leftarrow \mathcal{U}(\mathcal{S}_n)} [\mathcal{A}(\Lambda_{p,r}, R) = 1] \right| \leq \varepsilon.$$

2 Preliminary

2.1 Security Definition

In the rest of the paper, let $\mathcal{S}_n$ denote a set of all permutations of a set $\{1, \ldots, n\} =: [n]$.

We would like to model an adversary whose goal is to distinguish a permutation which is a result of PRPG algorithm from a permutation sampled from uniform distribution.

PRPG Algorithm. The PRPG algorithm starts with identity permutation of n elements π_0. In each round PRPG has access to a portion of n bits of the key stream (*i.e.,* in round l reads a portion of the key: $\mathcal{K}_l \in \{0,1\}^n$).

Leakage. We consider adversaries $\mathcal{A}$ which in the lth round can learn a fraction p of $\mathcal{K}_l$, namely $\Lambda_{p,l}(\mathcal{K}) = f_{l,p}(\mathcal{K}_l)$, where a function $f_{l,p}$ has range $\{*, \square\}^n$. More precisely, $f_{l,p} = a_1 a_2 \dots a_n$ where $a_i \in \{*, \square\}$. If $a_i = \square$ then the adversary sees the corresponding bit of the key stream (learns $\mathcal{K}_{l,i}$) and if $a_i = *$ then the adversary does not learn the bit at position i.

Example 1 (Adversary view). Let $\mathcal{K}_3 = 101110$ and $f_3 = 110100 = \square\square * \square * *$ then adversary's view is: $\Lambda_3 = \boxed{1}\boxed{0} * \boxed{1} * *$. Which means that the adversary learns that $\mathcal{K}_{3,1} = 1, \mathcal{K}_{3,2} = 0, \mathcal{K}_{3,4} = 1$.

We restrict our analysis only to the adversaries for which each bit can be eavesdropped independently with probability p, *i.e.,* $1 - P(a_i = *) = P(a_i = \square) = p$. This means that number of leaking bit has $Bin(n,p)$ distribution in each round (np bits are leaking in each round on average).

Let $\texttt{view}_r$ denote the view of the adversary at the end of the algorithm:

$$\texttt{view}_{r,p}(\mathcal{K}) = [\Lambda_{1,p}, \dots, \Lambda_{r,p}].$$

The distinguishability game for the adversary is as follows:

Definition 1. *The LEAK indistinguishability experiment* $\mathsf{Shuffle}^{LEAK}_{S,\mathcal{A}}(n,p,r)$*:*

1. S *is initialized with:*
   (a) *a key generated uniformly at random* $\mathcal{K} \sim \mathcal{U}(\{0,1\}^{rn})$,
   (b) $S_0 = \pi_0$ *(identity permutation).*
2. S *is run for* r *rounds:* $S_r := S(\mathcal{K})$ *and produces a permutation* π_r.
3. *Adversary obtains leaked bits of the key* $\mathsf{view}_{r,p}(\mathcal{K})$.
4. *We set:*
   - $c_0 := \pi_{rand}$ *a random permutation from uniform distribution is chosen,*
   - $c_1 := \pi_r$.
5. *A challenge bit* $b \in \{0,1\}$ *is chosen at random, permutation* c_b *is sent to the Adversary.*
6. *Adversary replies with* b'.
7. *The output of the experiment is defined to be* 1 *if* $b' = b$*, and* 0 *otherwise.*

In the case when adversary wins the game (if $b = b'$) we say that $\mathcal{A}$ succeeded. Adversary wins the game if she can distinguish the random permutation from the permutation being a result of the PRPG algorithm based on the leakage she saw.

Definition 2. *A shuffling algorithm* S *generates indistinguishable permutations in the presence of leakage if for all adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$Pr\left[\mathsf{Shuffle}_{\mathsf{S},\mathcal{A}}^{\mathsf{LEAK}}(n,p,r)=1\right] \leq \frac{1}{2}+\mathsf{negl}(n),$$

The above translates into:

Definition 3. *A shuffling algorithm* S *generates indistinguishable permutations in the presence of leakage if for all adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\left| \underset{K\leftarrow\{0,1\}^{keyLen}}{Pr}[\mathcal{A}(\Lambda_r,\mathsf{S}(K))=1] - \underset{R\leftarrow\mathcal{U}(\mathcal{S}_n)}{Pr}[\mathcal{A}(\Lambda_r,R)=1] \right| \leq \mathsf{negl}(n).$$

2.2 Markov Chains and Rate of Convergence

Consider ergodic Markov chain $\{X_k, k\geq 0\}$ on finite state space $\mathbb{E}=\{0,\ldots,M-1\}$ with stationary distribution ψ. Let $\mathcal{L}(X_k)$ denote the distribution of a chain at time k. Stating some results about the *rate of convergence* of a chain to its stationary distribution means having some knowledge on some distance *dist* (or a bound on it) between $\mathcal{L}(X_k)$ and ψ. By mixing time we mean the value of k making *dist* small, since it depends on the measure of the distance we define it as

$$\tau_{mix}^{dist}(\varepsilon)=\inf\{k : dist(\mathcal{L}(X_k),\psi)\leq\varepsilon\}.$$

In our applications the state space is a set of permutations of $[n]$, *i.e.,* $\mathbb{E}:=\mathcal{S}_n$ (thus $|\mathbb{E}|=n!$) and stationary distribution is a uniform one on $\mathbb{E}$ (we denote $\psi=\mathcal{U}(\mathbb{E})$).

Typically in literature the mixing time is defined for *dist* being total variation distance, i.e.,

$$d_{TV}(\mathcal{L}(X_k),\mathcal{U}(\mathbb{E}))=\frac{1}{2}\sum_{\sigma\in\mathcal{S}_n}\left|Pr(X_k=\sigma)-Pr(\psi=\sigma)\right|,$$

which in our case is equivalent to:

$$d_{TV}(\mathcal{L}(X_k),\mathcal{U}(\mathbb{E}))=\frac{1}{2}\sum_{\sigma\in\mathcal{S}_n}\left|Pr(X_k=\sigma)-\frac{1}{n!}\right|.$$

Note however that knowing that d_{TV} is small for some k does not imply that $\left|Pr(X_k=\sigma)-\frac{1}{n!}\right|$ are "uniformly" small, *i.e.,* that it is of order $1/n!$. This is very important observation, since it means that $\tau_{mix}^{d_{TV}}(\varepsilon)$ is not an adequate measure of mixing time for our applications (i.e., indistinguishability given in Definition 2). Instead we consider so-called **separation distance** defined by

$$sep(\mathcal{L}(X_k),\mathcal{U}(\mathbb{E})):=\max_{\sigma\in\mathbb{E}}\left(1-\frac{Pr(X_k=\sigma)}{Pr(\psi=\sigma)}\right)$$

which is:

$$sep(\mathcal{L}(X_k),\mathcal{U}(\mathbb{E})) := \max_{\sigma\in\mathbb{E}}(1 - n! \cdot Pr(X_k = \sigma))$$

If $sep(\mathcal{L}(X_k),\mathcal{U}(\mathbb{E})) \leq \varepsilon$ for some k (i.e., we know $\tau^{sep}_{mix}(\varepsilon)$), then

$$\left| Pr(X_k = \sigma) - \frac{1}{n!} \right| \leq \frac{\varepsilon}{n!}, \tag{1}$$

what will be crucial for showing our results.

Strong Stationary Times. The definition of separation distance fits perfectly into notion of Strong Stationary Time (SST) for Markov chains. This is a probabilistic tool for studying the rate of convergence of Markov chains.

Definition 4. *Random variable T is **Strong Stationary Time (SST)** if it is randomized stopping time for chain $\{X_k, k \geq 0\}$ such that:*

$$\forall(i \in \mathbb{E})\ Pr(X_k = i|T = k) = \psi(i).$$

Having SST T for chain with uniform stationary distribution lets us bound the separation distance (cf. [3])

$$sep(\mathcal{L}(X_k),\mathcal{U}(\mathbb{E})) \leq Pr(T > k). \tag{2}$$

We say that T is an optimal SST if $sep(\mathcal{L}(X_k),\mathcal{U}(\mathbb{E})) = Pr(T > k)$.

Remark. It is easy to show that separation distance is an upper bound on total variation distance, i.e. that $d_{TV}(\mathcal{L}(X_k),\mathcal{U}(\mathbb{E})) \leq sep(\mathcal{L}(X_k),\mathcal{U}(\mathbb{E}))$.

3 *RiffleSST*– Leakage Resilient Shuffle

3.1 General Pseudo-random Permutation Generator

We also model some leakage of information (to be specified later). We identify elements of $[n]$ with cards. We consider the following general pseudo-random permutation generator (PRPG) for generating a permutation of $[n]$. Initially we start with identity permutation π_0. At each round (step) we perform procedure `Shuffle` which takes the current permutation S and uses some "*randomness*" (based on secret key K) and updates the permutation. After r rounds the permutation is denoted by π_r. The algorithm stops depending on some stopping rule represented by procedure `StoppingRule` (Fig. 1).

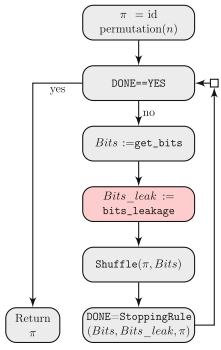


Fig. 1. General pseudo-random permutation generator (PRPG)

3.2 Description of *RiffleSST* Algorithm

Roughly speaking, it uses card shuffling scheme corresponding to time reversal of Riffle Shuffle (see [4]).

We do not specify here the details of `get_bits`, this should be a procedure which returns n bits from key K, can depend on current round number r, current permutation π, etc. (Fig. 2).

`RiffleShuffle` procedure performs the following: for given permutation of cards $\pi \in \mathcal{S}_n$ and given $Bits[i], i = 1, \ldots, n$ (think of assigning bit $Bits[i]$ to card on position i) we put all the cards with assigned bit 0 to the top *keeping* their relative ordering. Sample execution is given in Fig. 4: For example, for initial permutation $(1, 2, 3, 4, 5, 6)$ we assign bit 0 to cards $1, 2$ and 4, whereas we assign bit 1 to cards 3, 5 and 6. Thus, the resulting permutation is $(1, 2, 4, 3, 5, 6)$.

RiffleSST

```
get_bits      := (not specified)
bits_leakage  := get_bits_leakage_indep
Shuffle       := RiffleShuffle
StoppingRule  := StoppingRuleRiffle
```

Fig. 2. Leakage resilient shuffle *RiffleSST* algorithm.

---

procedure RIFFLESHUFFLE
  Input permutation π, round r, $Bits$ (of length n)
  Output updated permutation π

  s0:=1
  s1:=**sum**($Bits$)
  tmp=**vector**(n)
  for $i := 0$ to $n - 1$ **do**
    card=S[i]
    if($Bits[i]$=1) **do** tmp[s1]=card; s1=s1+1
    else tmp[s0]=card; s0=s0+1
    end if
  end for
  π:=tmp
end procedure

Leakage Model. We assume that at each step and at each position i (independently) a value of $Bit[i]$ is leaking with probability p. Function bits_leakage(p, n) generates n dimensional vector of zeros and ones. We assume that each coordinate is chosen independently being 1 with probability p and 0 with the

remaining probability. Note that the number of leaking bits has $Bin(n,p)$ distribution, thus on average np bits are leaking. The leakage is modeled by `get_bits_leakage_indep` procedure (here $unif(0,1)$ denotes a random variable uniformly distributed on $[0,1]$).

---

procedure GET_BITS_LEAKAGE_INDEP
  Input n, p
  Output vector $leak$ of n bits
  for $j = 1$ **to** n **do**
    $leak(j) =$**1**$\{unif(0,1) < p\}$
  end forreturn $leak$
end procedure

We simply run the algorithm for pre-defined number of steps expressed by procedure `StoppingRuleRiffle`.

---

procedure StoppingRuleRiffle
  Input n, p (leakage level), ε
  Output {YES,NO}

  if $r < \log_{\frac{2}{2-(1-p)^2}} \binom{n}{2} + \log_{\frac{2}{2-(1-p)^2}} \left(\frac{1}{\varepsilon n!}\right)$ **then return** NO
  else return YES
  end if
end procedure

---

4 Proofs

As already mentioned, assuming random keys, the algorithm can be regarded as (time reversal of) Riffle Shuffle scheme. The idea is similar to approach presented in [13], following author's notation we will call a version of the algorithm with random keys as *idealized* one. Showing that after the execution of the algorithm the adversary has no non-negligible knowledge (in both, leakage and no-leakage version) corresponds to showing that after shuffling cards as many times as the number of steps of the algorithm, the resulting permutation is close to uniform one. In other words, proving the theorem reduces to studying the rate of convergence of corresponding Markov chains. However, the typical bounds on the rate of convergence involving total variation distance do not imply that the shuffling algorithm generates permutation which is indistinguishable from random permutation according to Definition 2. That is why we focus on bounds for separation distance what is achieved by using Strong Stationary Times technique.

Consider the *idealized* version of *RiffleSST* (call it `RiffleSST*`) which is defined by the specification from Fig. 4. Roughly speaking, there are two differences compared to `RiffleSST`: (i) in each round we take new n random bits. (ii) instead of running it pre-defined number of steps, we use `StoppingRuleRifflePairs` as stopping rule. The stopping rule works as follows: Initially we set all $\binom{n}{2}$ pairs $(i,j), i,j = 1,\ldots,i<j$ as *not-marked*. (This can be simply represented as $\binom{n}{2}$-dimensional vector with all entries set to 0). Given the current permutation $\pi \in \mathcal{S}_n$ and $Bits[i], Bits\_leak[i], i = 1,\ldots,n$ we mark the pair (i,j) (or equivalently, we say that pair (i,j) is *updated*) if $(Bits[i] \oplus Bits[j] = 1)$ **and** $(Bits\_leak[i] \vee Bits\_leak[j] = 0)$ (*i.e.,* cards $S[i]$ and $S[j]$ were assigned different bits and none is leaking). Formally, this is given in `StoppingRuleRifflePairs` procedure (Fig. 3).

```
RiffleSST*

get_bits        := get_bits_rand
bits_leakage    := get_bits_leakage_indep
Shuffle         := RiffleShuffle
StoppingRule    := StoppingRuleRifflePairs
```

Fig. 3. Idealized version of leakage resilient shuffle *RiffleSST*

---

procedure StoppingRuleRifflePairs
  Input set of already updated $pairs(i,j), i<j$, $Bits, Bits\_leak$
  Output {YES,NO}

  for each pair (i,j) **do**
    if $(Bits[i] \oplus Bits[j] = 1)$ **and** $(Bits\_leak[i] = Bits\_leak[j] = 0)$ **then**
      mark pair (i,j)
    end if
  end for
  if all $\binom{n}{2}$ pairs are marked **then return** YES
  elsereturn NO
  end if
end procedure

The main ingredients of the proof of Theorem 1 are the following Lemma 1 and Theorem 2.

Lemma 1. *The resulting permutation of RiffleSST* has a uniform distribution over* $\mathcal{S}_n$.

Proof (of Lemma 1). For leakage level $p = 0$ the procedure `RiffleSST*` is exactly the Markov chain corresponding to *time-reversed Riffle Shuffle* card shuffling. At each step we consider all $\binom{n}{2}$ pairs (i,j) and we "mark" each pair if either card i was assigned 0 and card j was assigned 1, or vice-versa. Since these two events have equal probability, thus relative ordering of these two cards is random at such step. Let T be the first time all pairs are "marked". Then all the pairs are in

relative random order and thus the permutation is also random. In other words, the distribution of X_k given $T = k$ is uniform. This means that the running time T of the algorithm is a Strong Stationary Time for riffle shuffle procedure and the distribution of the chain at time T is uniform.

Note that we exchangeably use the term "mixed" and "updated".

For general $p \in [0, 1)$ the situation is not much different: We update only pairs which are assigned different bits and such that both cards are not leaking. Thus, once the pair is updated it means that it is in random relative order and adversary has no knowledge about this order. After updating all the pairs she has no knowledge about relative order of all the pairs, thus, from her perspective, resulting permutation is random. We note that the knowledge about the permutation can already "vanish" earlier (*i.e.,* before updating all the pairs), but it is for sure that time till updating all the pairs is enough.

Note that the no leakage version (*i.e.,* when $p = 0$) of the algorithm can be written in a more compact way (similarly as in [4], the pairs are not involved there directly), however this notation lets us relatively easy extend the algorithm into a leakage resilient version.

Theorem 2. *Let $\{X_k\}_{k\geq 0}$ be the chain corresponding to RiffleSST*. The mixing time τ_{mix}^{sep} of the chain is given by*

$$\tau_{mix}^{sep}(\varepsilon) \leq \log_{\frac{2}{2-(1-p)^2}} \binom{n}{2} + \log_{\frac{2}{2-(1-p)^2}} \left(\varepsilon^{-1}\right).$$

Proof (of Theorem 2). In Lemma 1 we showed that $T := \inf_k\{X_k = 0\}$ (*i.e.,* first moment when all pairs are updated) is an SST.

Let T_{ij} be the first time when cards i and j are updated. In one step, the probability that given pair will not be updated is $1 - \frac{1}{2}(1-p)^2 = \frac{2-(1-p)^2}{2}$. We have

$$\begin{aligned} Pr(T > k) &= Pr\left(\bigcup_{1\leq i<j\leq n} \{T_{ij} > k\}\right) \leq \sum_{1\leq i<j\leq n} Pr(T_{ij} > k) \\ &= \textstyle\sum_{1\leq i<j\leq n} \left(1 - (1-p)^2 \cdot \frac{1}{2}\right)^k = \binom{n}{2} \left(\frac{2-(1-p)^2}{2}\right)^k. \end{aligned}$$

For $k = \log_{\frac{2}{2-(1-p)^2}} \binom{n}{2} + \log_{\frac{2}{2-(1-p)^2}} \left(\varepsilon^{-1}\right)$ we have $Pr(T > k) \leq \varepsilon$, using (2) finishes the proof.

In Theorem 1 we perform *RiffleSST* for $r = \tau_{mix}^{sep}(\varepsilon n!)$ steps, what means that separation distance is less or equal to $\varepsilon n!$. From (1) we thus have that $\left|Pr(X_r = \sigma) - \frac{1}{n!}\right| \leq \varepsilon$ for any permutation σ. This together with Lemma 1 and Theorem 2 completes the proof of Theorem 2.

Remark 1. Note that if we replace τ_{mix}^{sep} with $\tau_{mix}^{d_{TV}}$ in Theorem 2, we could not conclude Theorem 1. This is because knowing that separation distance is smaller than ε is much stronger than knowing that total variation distance is smaller

than ε, in particular (1) holds. (Note that typically, *e.g.,* coupling methods provide directly bounds on total variation distance). However, knowing that total $d_{TV}(\mathcal{L}(X_k),\mathcal{U}(\mathbb{E})) \leq \varepsilon$ implies (under some mild conditions - see Theorem 7 in [2]) that $sep(\mathcal{L}(X_{2k}),\mathcal{U}(\mathbb{E})) \leq \varepsilon$ what means that twice as many steps would be needed to achieve security claimed in Theorem 1.

5 Sample Execution of *RiffleSST* Algorithm

In Fig. 4 the sample execution with and without leakage is presented. At each step, the left column represents the current permutation, whereas the right one currently assigned bits. The leaking bits are represented by red-shaded boxes. In Fig. 5 updated pairs for leakage and no leakage versions are given.

For example:

- *No leakage version*: At step 3. the current permutation is $(1, 4, 6, 2, 3, 5)$ and assigned bits are $Bits = (1, 0, 1, 1, 1, 0)$. Thus, *e.g.,* card on position 1 (1) and

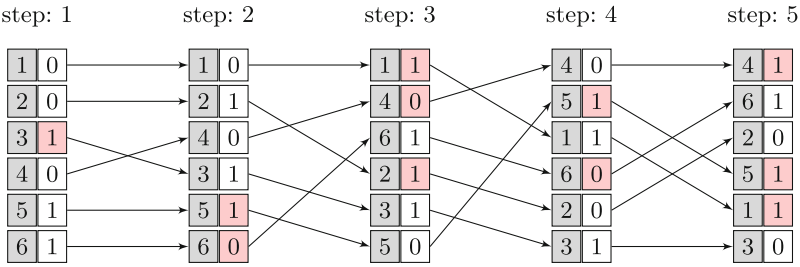


Fig. 4. Sample execution of our PRPG `RiffleSST`* algorithm for $n = 6$. Current permutation at each step is the left column (grayed) whereas right column are the chosen bits. Red shaded bits are leaking. (Color figure online)

	step: 1	step: 2	step: 3	step: 4	step: 5
No leakage	**(1,3), (1,5),(1,6)** **(2,3), (2,5), (2,6)** **(3,4), (4,5), (4,6)** **sum**$(pairs)$=9	**(1,2)**, (1,3), (1,5) **(2,4)**,(2,6) (3,4), **(3,6)**, (4,5) **(5,6)** **sum**$(pairs)$=13	**(1,4)**, (1,5) (2,4), (2,6) (3,4), **(3,5)**, (5,6) **sum**$(pairs)$=15 STOP		
Leakage	**(1,5), (1,6)** **(2,5), (2,6)** **(4,5), (4,6)** **sum**$(pairs)$=6	**(1,2), (1,3)** **(2,4)** **(3,4)** **sum**$(pairs)$=10	**(3,5), (5,6)** **sum**$(pairs)$=12	(1,2),**(1,4)** **(2,3)** (3,4) **sum**$(pairs)$=14	(2,6) **(3,6)** **sum**$(pairs)$=15 STOP

Fig. 5. Pairs "*mixed*" at each step of execution of PRPG given in Fig. 4. New pairs are **bolded**. The idealized algorithm (both, in non leakage and leakage version) stops when $\binom{6}{2} = 15$ pairs are mixed.

card on position 2 (4) have different bits assigned (respectively 1 and 0), thus this pair **(1,4)** is updated (it is bolded since it is first step when different bits were assigned for this pair).

- *Leakage version*: At the same step 3. bits assigned to cards 1, 4 and 2 are leaking. Thus all the pairs involving any of these cards are not considered, what results only in updating pairs **(3,5), (5,6)**.

6 Conclusions

We presented the first analysis of the rate of convergence of the riffle-shuffle in the presence of leakage. We proved that no adversary can distinguish permutations produced by the *RiffleSST* (after enough number of steps – Theorem 1) from permutations sampled from uniform distribution.

Open Problems. Since the number of permutations is of $[n]$ is $n!$ the entropy of the uniform distribution on $[n]$ is $O(n \log n)$. The time-reversed riffle-shuffle is optimal (up to a constant factor) shuffle since its mixing-time is $O(\log n)$ and each round consumes n bits of randomness, so in total $O(n \log n)$ bits are used. In case of no leakage ($p = 0$) the mixing time of *RiffleSST* * - by Theorem 2 - is $O(n \log n)$ and thus is optimal (Fig. 6).

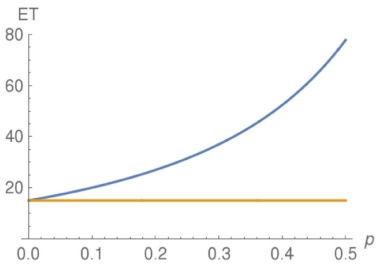


Fig. 6. Comparison of the expected number of rounds for $n = 256$ and Riffle-Shuffle (without leakage – orange) and *RiffleSST* with the leakage level p – blue. (Color figure online)

Question: Is `RiffleSST`* optimal in the presence of leakage with rate $p > 0$? More precisely, can we use fewer bits than $O\left(n \log_{\frac{2}{1-(1-p)^2}} n\right)$ bits to achieve the security claimed in Theorem 1?

References

1. Albrecht, M.R., Paterson, K.G.: Lucky microseconds: a timing attack on Amazon's *s2n* implementation of TLS. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 622–643. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_24
2. Aldous, D., Diaconis, P.: Shuffling cards and stopping times. Am. Math. Mon. **93**(5), 333–348 (1986)
3. Aldous, D., Diaconis, P.: Strong uniform times and finite random walks. Adv. Appl. Math. **97**, 69–97 (1987)
4. Diaconis, P., Shahshahani, M.: Generating a random permutation with random transpositions. Zeitschrift fur Wahrscheinlichkeitstheorie und Verwandte Gebiete **57**(2), 159–179 (1981)
5. Genkin, D., Pachmanov, L., Pipman, I., Tromer, E.: Stealing keys from PCs using a radio: cheap electromagnetic attacks on windowed exponentiation. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 207–228. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_11

6. Genkin, D., Shamir, A., Tromer, E.: RSA key extraction via low-bandwidth acoustic cryptanalysis. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 444–461. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_25
7. Gomułkiewicz, M., Klonowski, M., Kutyłowski, M.: Rapid mixing and security of Chaum's visual electronic voting. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 132–145. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39650-5_8
8. Hoang, V.T., Morris, B., Rogaway, P.: An enciphering scheme based on a card shuffle. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 1–13. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_1
9. Jakobsson, M., Juels, A., Rivest, R.: Making mix nets robust for electronic voting by randomized partial checking. In: USENIX Security Symposium (2002)
10. Krzywiecki, Ł., Kutyłowski, M., Misztela, H., Strumiński, T.: Private information retrieval with a trusted hardware unit – revisited. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt 2010. LNCS (LNAI and LNBI), vol. 6584, pp. 373–386. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21518-6_26
11. Kulis, M., Lorek, P., Zagorski, F.: Randomized stopping times and provably secure pseudorandom permutation generators. In: Phan, R.C.-W., Yung, M. (eds.) Mycrypt 2016. LNCS, vol. 10311, pp. 145–167. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61273-7_8
12. Lorek, P., Zagórski, F., Kulis, M.: Strong stationary times and its use in cryptography. IEEE Trans. Dependable Secure Comput. 1–14 (2017)
13. Mironov, I.: (Not so) Random shuffles of RC4. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 304–319. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_20
14. Morris, B., Rogaway, P., Stegers, T.: How to encipher messages on a small domain. Deterministic encryption and the thorp shuffle. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 286–302. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_17
15. Morris, B., Rogaway, P.: Sometimes-recurse shuffle. Almost-random permutations in logarithmic expected time. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 311–326. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_18
16. Naor, M., Reingold, O.: On the construction of pseudo-random permutations. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing - STOC 1997, New York , USA, pp. 189–199. ACM Press (1997)
17. Ristenpart, T., Yilek, S.: The mix-and-cut shuffle: small-domain encryption secure against N queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 392–409. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_22
18. Schuldt, J.C.N., Rivest, R.L.: Spritz–a spongy RC4-like stream cipher and hash function. Technical report (2014)
19. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_3
20. Standaert, F.-X., Pereira, O., Yu, Y.: Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS (LNAI and LNBI), vol. 8042, pp. 335–352. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_19

21. Standaert, F.-X., Pereira, O., Yu, Y., Quisquater, J.-J., Yung, M., Oswald, E.: Leakage resilient cryptography in practice. In: Sadeghi, A.R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security. ISC, pp. 99–134. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14452-3_5
22. Yang, Y., Ding, X., Deng, R.H., Bao, F.: An efficient PIR construction using trusted hardware. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 64–79. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85886-7_5
23. Yarom, Y., Genkin, D., Heninger, N.: CacheBleed: a timing attack on OpenSSL constant time RSA. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 346–367. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53140-2_17

Ordinary Pairing-Friendly Genus 2 Hyperelliptic Curves with Absolutely Simple Jacobians

Georgios Fotiadis(✉) and Elisavet Konstantinou

Department of Information and Communication Systems Engineering,
University of the Aegean, 83200 Karlovassi, Samos, Greece
{gfotiadis,ekonstantinou}@aegean.gr

Abstract. We present a method for producing pairing-friendly, simple, ordinary Jacobian varieties of genus 2 hyperelliptic curves defined over a prime field $\mathbb{F}_p$. The proposed method heavily relies on the construction of a suitable p-Weil number and a corresponding quartic CM-field. Our Jacobians are absolutely simple and for this special class of Jacobians we give the first examples in the literature with ρ-values below 4, while previous results had in general ρ-values between 6 and 8. These examples derive from "families" of pairing-friendly Jacobians, which are basically polynomial representations of the Jacobian parameters.

Keywords: Pairing · Hyperelliptic curves · Jacobian
Embedding degree

1 Introduction

An *asymmetric pairing* is a bilinear, non-degenerate, efficiently computable map $\widehat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_{\mathrm{T}}$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathrm{T}}$ are cyclic groups of prime order r with $\mathbb{G}_1 \neq \mathbb{G}_2$. A crucial cryptographic requirement is that the discrete logarithm problem (DLP) is computationally infeasible in all *pairing groups* $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathrm{T}}$. We call $\mathbb{G}_1, \mathbb{G}_2$ the *source groups* and $\mathbb{G}_{\mathrm{T}}$ the *target group*. Initially the source groups were set as r-order subgroups of ordinary elliptic curves over a finite field, while the target group was an r-order subgroup of a finite field.

Since elliptic curves are genus 1 algebraic curves, an obvious question is whether pairings on higher genus curves can be also used in implementations. In this case $\mathbb{G}_1$ and $\mathbb{G}_2$ consist of elements in the Jacobian variety of a genus g hyperelliptic curve defined over a finite field. By [2,19], this can be an advantageous choice especially when $g = 2$, since genus 2 curves and their Jacobians:

1. are competitive to elliptic curves in performance and security [2,19].
2. result in efficient Tate pairing calculations [11].
3. have efficient CM-constructions [20,25] and point operations [4].
4. have points with smaller size.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 409–424, 2017.
https://doi.org/10.1007/978-3-319-72453-9_33

This is our motivation for constructing "pairing-friendly", ordinary Jacobians of genus 2 hyperelliptic curves over a prime field $\mathbb{F}_p$, called the *base field*.

An affine genus 2 hyperelliptic curve C over $\mathbb{F}_p$ is defined by the equation $C/\mathbb{F}_p : y^2 = F(x)$, where $F(x) \in \mathbb{F}_p[x]$ is monic with $\deg F \in \{5, 6\}$. For any extension $\Bbbk$ of $\mathbb{F}_p$, we denote by $C(\Bbbk)$ the set of all points with coordinates in $\Bbbk$ satisfying the hyperelliptic curve equation. Unlike the genus 1 case this set is not a group and hence we cannot define DLP-based protocols on $C(\Bbbk)$. However, to each such curve we associate a special object called the Jacobian [18] of $C/\mathbb{F}_p$, denoted by $J(\mathbb{F}_p)$. This is a 2-dimensional abelian variety, hence an algebraic group, with order $\#J(\mathbb{F}_p) \approx p^2$. The elements of $J(\mathbb{F}_p)$ are equivalence classes of zero degree divisors, defined over $\mathbb{F}_p$, under the linear equivalence of divisors (see Sect. 2). This can be generalized to any extension $\Bbbk$ of $\mathbb{F}_p$. In our context we assume that $J(\mathbb{F}_p)$ contains a cyclic subgroup of prime order r and that it is ordinary, simple and absolutely simple [21] (see also Sect. 2).

For asymmetric pairings on Jacobians, the source groups are distinct r-order subgroups of $J(\mathbb{F}_{p^k})$ and the target group is an r-order subgroup of the multiplicative group of the extension field $\mathbb{F}_{p^k}$. In other words a pairing maps two divisors of order r, defined over $\mathbb{F}_{p^k}$, to an rth root of unity. This positive integer k is called the *embedding degree of* $J(\mathbb{F}_p)$ *with respect to* r and it is the smallest positive integer such that $\mathbb{F}_{p^k}$ contains all rth roots of unity. In pairing-based applications such Jacobians are chosen according to the following rules:

1. The order of the Jacobian has a large prime factor r, i.e. $\#J(\mathbb{F}_p) = hr$, for $h \geq 1$. This ensures that $J(\mathbb{F}_p)$ (hence $J(\mathbb{F}_{p^k})$) contains points of order r.
2. The prime r is large enough, so that the DLP in $\mathbb{G}_1, \mathbb{G}_2$ is computationally hard. According to today's requirements, r should be at least 256 bit large, to avoid Pollard's rho attack, with running time $O(\sqrt{r})$.
3. The embedding degree k is large enough, so that the DLP in $\mathbb{G}_\mathrm{T} \subset \mathbb{F}_{p^k}^*$ is as hard as in $\mathbb{G}_1, \mathbb{G}_2$. In practice $\mathbb{F}_{p^k}$ must be resistant to the variants of the number field sieve (NFS) attack [7,14,17].
4. k is relatively small, for efficient operations in $\mathbb{G}_\mathrm{T}$. This means that the extension field must be as large as to ensure security and no larger.
5. The ρ-value $\rho = 2 \log p / \log r$ of the Jacobian is close to 1. This saves bandwidth by keeping the representation of Jacobian elements small. Examples with $\rho \approx 1$ are still absent for ordinary, absolutely simple Jacobians.

Hyperelliptic curves and the corresponding Jacobians satisfying these properties are called *pairing-friendly*.

In this paper we describe a method for producing pairing-friendly ordinary Jacobians of genus 2 hyperelliptic curves defined over prime fields. We present new examples of absolutely simple Jacobians, with the best reported ρ-values so far in the literature, for various embedding degrees. Particularly, our examples reduce the ρ-value to be up to 4, while previous results for the same embedding degrees have in general ρ-values between 6 and 8 [8], or around 8 [20].

In Sect. 2 we present the necessary background for pairing-friendly 2-dimensional Jacobians and a summary of methods for their construction. We

analyze our proposal and demonstrate our recommendations in Sect. 3. Numerical results of cryptographic value are provided in Sect. 4 and we conclude the paper in Sect. 5, summarizing our recommendations.

2 Background

Genus 2 Hyperelliptic Curves and Jacobians. Let C be a genus 2 hyperelliptic curve over a prime field $\mathbb{F}_p$ and $C(\Bbbk)$ the set of points on the curve with coordinates in an extension $\Bbbk$ of $\mathbb{F}_p$. Since $C(\Bbbk)$ is not a group, in hyperelliptic curve cryptography we are working with the Jacobian $J(\mathbb{F}_p)$ of $C/\mathbb{F}_p$ [18], which is a 2-dimensional abelian variety and hence an algebraic group [21]. It is also a quotient group, whose elements are equivalence classes of zero degree divisors under the linear equivalence of divisors. In particular, two zero degree divisors are linearly equivalent, if their difference is a principal divisor, i.e. a divisor of a rational function in the function field of the curve $C/\mathbb{F}_p$ [21,23]. In dimension 2, each equivalence class consists of exactly two elements.

In this paper we are working with *simple* Jacobians which are also *absolutely simple*. A 2-dimensional Jacobian is simple if it does not split over $\mathbb{F}_p$ to a product of elliptic curve groups and it is absolutely simple, if it remains simple over $\overline{\mathbb{F}}_p$ [21]. We denote by $\mathrm{End}(J(\mathbb{F}_p))$ the *endomorphism ring* containing all homomorphisms from $J(\mathbb{F}_p)$ to itself. One of these elements is the *Frobenius endomorphism*, denoted by π, which acts by raising a divisor in $J(\mathbb{F}_p)$ to the pth power. When $J(\mathbb{F}_p)$ is simple, the Frobenius endomorphism satisfies a quartic, monic polynomial $P(x) \in \mathbb{Z}[x]$ called the *characteristic polynomial of Frobenius*:

$$P(x) = \prod_{i=1}^{4} [x - \sigma_i(\pi)] = x^4 + Ax^3 + Bx^2 + Apx + p^2, \tag{1}$$

where σ_i are the embeddings of the number field $K = \mathbb{Q}(\pi)$ into $\mathbb{C}$. Thus, π is an algebraic integer and also a *p-Weil number*, meaning $\pi\overline{\pi} = p$, where $\overline{\pi}$ is the complex conjugate of π. In our case, $J(\mathbb{F}_p)$ will be ordinary and K a quartic *CM-field*, i.e. an imaginary quadratic extension of a totally real field [21].

The order of the Jacobian and $P(x)$ are related by $\#J(\mathbb{F}_p) = P(1)$ [5]. Additionally, $J(\mathbb{F}_p)$ is ordinary if $\gcd(B, p) = 1$ [13] and it is simple if $P(x)$ is irreducible over $\mathbb{Z}[x]$ [23]. Finally, in order to check if $J(\mathbb{F}_p)$ is absolutely simple we use the next fact [13].

Proposition 1. Let $J(\mathbb{F}_p)$ be a 2-dimensional Jacobian, with characteristic polynomial of Eq. (1). Then exactly one of the following holds: (1) $J(\mathbb{F}_p)$ is absolutely simple. (2) $A = 0$. (3) $A^2 = p + B$. (4) $A^2 = 2B$. (5) $A^2 = 3B - 3p$. In cases (2), (3), (4) and (5), the smallest extension of $\mathbb{F}_p$ over which $J(\mathbb{F}_p)$ splits, is quadratic, cubic, quartic and sextic respectively.

Proof. See Theorem 6, p. 145 in [13].

Pairing-Friendly Conditions. Recall that for asymmetric pairings on Jacobians, $\mathbb{G}_1, \mathbb{G}_2$ are distinct subgroups of $J(\mathbb{F}_{p^k})$, while $\mathbb{G}_T$ is an r-order subgroup of the multiplicative group of $\mathbb{F}_{p^k}$, where k is the embedding degree. This is the smallest positive integer such that $\mathbb{F}_{p^k}$ contains the group μ_r of rth roots of unity. Equivalently, it is the smallest positive integer, such that $r \mid (p^k - 1)$ [8].

Freeman et al. [9] described the conditions for g-dimensional Jacobians to have embedding degree k. Here we are restricted to $g = 2$.

Proposition 2. Let $J(\mathbb{F}_p)$ be an ordinary 2-dimensional Jacobian with Frobenius endomorphism π and characteristic polynomial of Frobenius $P(x) \in \mathbb{Z}[x]$. Let k be a positive integer and $\Phi_k(x)$ the kth cyclotomic polynomial and suppose that $\gcd(r, p) = 1$ and $K = \mathbb{Q}(\pi)$ is a quartic CM-field. If

$$\#J(\mathbb{F}_q) = P(1) \equiv 0 \bmod r \qquad \text{and} \qquad \Phi_k(p) \equiv 0 \bmod r, \tag{2}$$

then $J(\mathbb{F}_p)$ has embedding degree k with respect to r.

Proof. See Proposition 2.1 in [9].

Thus, in order to construct ordinary and simple 2-dimensional Jacobians over $\mathbb{F}_p$ with embedding degree k and an r-order subgroup, it suffices to search for a Frobenius endomorphism $\pi \in \mathrm{End}(J(\mathbb{F}_p))$ and a quartic CM-field $K = \mathbb{Q}(\pi)$, such that System (2) is satisfied. Note that the second equation in System (2) implies that p is a primitive kth root of unity in $(\mathbb{Z}/r\mathbb{Z})^*$.

As stated in Sect. 1, r must be a large prime so that the DLP in the r-order subgroups $\mathbb{G}_1, \mathbb{G}_2 \subseteq J(\mathbb{F}_{p^k})$ is computationally hard and the embedding degree k must be large enough so that the DLP in $\mathbb{G}_T \subseteq \mathbb{F}^*_{p^k}$ is approximately of the same difficulty as in $\mathbb{G}_1, \mathbb{G}_2$. Note that k should be the smallest such integer, since the extension field $\mathbb{F}_{p^k}$ must not be unnecessarily large. The ideal case appears when $\#J(\mathbb{F}_p)$ and r have approximately the same size. Since $\#J(\mathbb{F}_p) \approx p^2$, this means that the ρ-value $\rho = 2 \log p / \log r$ must be close to 1 [9]. The recommended sizes of Jacobian parameters and the security levels that they provide are discussed in Sect. 4 (see also [1]). The simple and ordinary Jacobians having the properties we studied in this paragraph are called *pairing-friendly* [8].

Parametric Families. The most common way to produce pairing-friendly Jacobians is to represent its parameters as polynomials, which when evaluated at certain integers will produce the actual Jacobian parameters. This idea was first introduced by Brezing and Weng [3] for elliptic curves and generalized by David Freeman [8] for higher dimensional abelian varieties. In this case the Frobenius endomorphism is represented by a polynomial $\pi(x) \in K[x]$ with characteristic polynomial of Frobenius $P(t) \in \mathbb{Q}[t]$:

$$P(t) = \prod_{i=1}^{4} [t - \sigma_i(\pi(x))] = t^4 + A(x)t^3 + B(x)t^2 + A(x)pt + p^2, \tag{3}$$

for the four embeddings $\sigma_i : K \longrightarrow \mathbb{C}$ and some $A(x), B(x) \in \mathbb{Z}[x]$. Such a polynomial representation allows us to work with *polynomial families of pairing-friendly Jacobians*. The precise definition is the following [8].

Definition 1. Let K be a quartic CM-field, $\pi(x) \in K[x]$ and $r(x) \in \mathbb{Q}[x]$. The pair $[\pi(x), r(x)]$ *parametrizes a family of pairing-friendly Jacobians with embedding degree* k, if the following conditions are satisfied:

1. $p(x) = \pi(x)\overline{\pi}(x) \in \mathbb{Q}[x]$ and $p(x)$ represents primes.
2. $r(x)$ is non-constant, irreducible, integer-valued, with $\mathrm{lc}(r) > 0$.
3. $P(1) \equiv 0 \bmod r(x)$.
4. $\Phi_k(p(x)) \equiv 0 \bmod r(x)$, where $\Phi_k(x)$ is the kth cyclotomic polynomial.

By saying that $p(x)$ represents primes we mean that it is non-constant, irreducible, with $\mathrm{lc}(p) > 0$ and it returns primes for finitely (or infinitely) many $x \in \mathbb{Z}$ [8]. Condition (3) ensures that the Jacobian order factorizes as $\#J(\mathbb{F}_p) = h(x)r(x)$, for some $h(x) \in \mathbb{Q}[x]$, while condition (4) implies that $p(x)$ is a primitive kth root of unity in $\mathbb{Q}[x]/\langle r(x)\rangle$. Although $r(x)$ can be chosen as any polynomial with rational coefficients satisfying condition (2) of Definition 1, it is usually considered as the kth cyclotomic polynomial. Finally, the ρ-value of a polynomial family $[\pi(x), r(x)]$ is defined as the ratio:

$$\rho(\pi, r) = \lim_{x\to\infty} \frac{2\log p(x)}{\log r(x)} = \frac{2\deg p}{\deg r}.$$

Previous Constructions. Methods for constructing absolutely simple Jacobians are given in [8,9,20], with ρ-value in the range $6 \leq \rho \leq 8$. However better ρ-values can be achieved by non-absolutely simple Jacobians. For example see [6,10,12,15,16], with generic $\rho \leq 4$, where the best results appear in [6], with $2 \leq \rho < 4$. Unfortunately there are still no examples with $\rho < 2$ for simple, ordinary Jacobians. All methods in [6,8,10,12,15,16] use polynomial families of pairing-friendly Jacobians. An alternative approach is presented by Lauter–Shang in [20]. Representing the Frobenius element $\pi \in K$ in an appropriate form, they derive a system of three equations in four variables, whose solutions lead to few examples of absolutely simple Jacobians with $\rho \approx 8$.

Contribution. In this paper we focus on pairing-friendly 2-dimensional, absolutely simple and ordinary Jacobians. Their construction depends mainly on the choice of the quartic CM-field K and the representation of the Frobenius endomorphism π. We present a procedure for constructing polynomial families of pairing-friendly Jacobians based on Lauter-Shang's [20], Dryło's [6] and new polynomial representations of the Frobenius endomorphism. In each case the problem of constructing the families is reduced to a system of three equations in four variables. By their solutions we produced polynomial families of 2-dimensional, absolutely simple Jacobians with the best ρ-values so far in the literature. In particular our families have in general $\rho(\pi, r) \leq 4$ for various embedding degrees, while previous results had $\rho(\pi, r)$ between 6 and 8. Using our families we produced various numerical examples of cryptographic value.

3 Constructing Pairing-Friendly Jacobians

Let $C/\mathbb{F}_p$ be a genus 2 hyperelliptic curve for some prime p, with a simple and ordinary Jacobian $J(\mathbb{F}_p)$ and suppose that $\#J(\mathbb{F}_p) = hr$, for some prime r, with $\gcd(r, p) = 1$ and $h > 0$. Let also k be a positive integer and K a quartic CM-field. We can determine suitable parameters of a 2-dimensional Jacobian by searching for a Frobenius element $\pi \in K$, such that System (2) is satisfied:

$$P(1) \equiv 0 \bmod r \qquad \text{and} \qquad \Phi_k(p) \equiv 0 \bmod r \iff p = \pi\overline{\pi} \equiv \zeta_k \bmod r, \tag{4}$$

where $P(x) \in \mathbb{Z}[x]$ is the characteristic polynomial of Frobenius given by Eq. (1) and ζ_k a primitive kth root of unity.

Since we will be working with polynomial families we need to transfer the above situation in terms of polynomial representations. This means that the Frobenius endomorphism is a polynomial $\pi(x) \in K[x]$, with characteristic polynomial of Frobenius $P(t) \in \mathbb{Z}[t]$ given by Eq. (3). The complete process for constructing polynomial families of pairing-friendly, 2-dimensional Jacobians is described in Algorithm 1. We first fix an integer $k > 0$, a quartic CM-field K and set L as the number field containing ζ_k and K. Usually L is taken as the lth cyclotomic field $\mathbb{Q}(\zeta_l)$ for some $l \in \mathbb{Z}_{>0}$, such that $k \mid l$. In step 1, we construct the polynomial $r(x)$ such that it satisfies condition (2) of Definition 1. If $L = \mathbb{Q}(\zeta_l)$, then $r(x) = \Phi_l(x)$. With this choice we know that the polynomial $u(x) = x$ is a primitive lth root of unity in $\mathbb{Q}[x]/\langle r(x)\rangle$. Then the primitive kth roots of unity can be obtained by computing the powers $u(x)^i \bmod r(x)$, for every $i = 1, \ldots, \varphi(l) - 1$, such that $l/\gcd(i, l) = k$. The fourth step is the most demanding since we are searching for the Frobenius polynomial $\pi(x) \in K[x]$, such that the family of Jacobians is pairing-friendly. To come to this conclusion we also need to verify that the polynomial $p(x) = \pi(x)\overline{\pi}(x)$ represents primes (step 5). The output of Algorithm 1 is a polynomial family $[\pi(x), r(x)]$ of pairing-

Algorithm 1. Constructing families of pairing-friendly 2-dimensional Jacobians.

Input: An integer $k > 0$, a quartic CM-field K, a number field L containing ζ_k, K.
Output: A polynomial family $[\pi(x), r(x)]$ of pairing-friendly, 2-dimensional Jacobian variety, with embedding degree k.

1: Find an $r(x) \in \mathbb{Q}[x]$ satisfying condition (2) of Definition 1, s.t. $L \cong \mathbb{Q}[x]/\langle r(x)\rangle$.
2: Let $u(x) \in \mathbb{Q}[x]$ be a primitive lth root of unity in $\mathbb{Q}[x]/\langle r(x)\rangle$.
3: For every $i = 1, \ldots, \varphi(l) - 1$, such that $l/\gcd(i, l) = k$, do the following:
4: Find a polynomial $\pi(x) \in K[x]$, satisfying the following System:

$$\#J(\mathbb{F}_p) = P(1) \equiv 0 \bmod r(x) \quad \text{and} \quad p(x) = \pi(x)\overline{\pi}(x) \equiv u(x)^i \bmod r(x) \tag{5}$$

5: If $p(x) = \pi(x)\overline{\pi}(x)$ represents primes return the family $[\pi(x), r(x)]$.

friendly 2-dimensional Jacobians with embedding degree k and ρ-value:

$$\rho(\pi, r) = \frac{2\deg p}{\deg r} = \frac{2(\deg \pi + \deg \overline{\pi})}{\deg r} \leq \frac{2(2\deg r - 2)}{\deg r} = 4 - \frac{4}{\deg r} < 4.$$

This is a significant improvement compared to [8,9,20], which for absolutely simple Jacobians have $6 \leq \rho(\pi, r) \leq 8$.

3.1 Lauter-Shang's Frobenius Elements

Lauter and Shang [20] considered quartic CM-fields $K = \mathbb{Q}(\eta)$, with positive and square-free discriminant Δ_K (primitive CM-fields), where η is:

$$\eta = \begin{cases} i\sqrt{a + b\sqrt{d}}, & \text{if} \quad d \equiv 2, 3 \bmod 4 \\ i\sqrt{a + b\dfrac{-1 + \sqrt{d}}{2}}, & \text{if} \quad d \equiv 1 \bmod 4 \end{cases} \tag{6}$$

for some $a, b, d \in \mathbb{Z}$, where d is positive and square-free. The Frobenius endomorphism π is an element of K and hence it is of the form:

$$\pi = X + Y\sqrt{d} + \eta\left(Z + W\sqrt{d}\right), \tag{7}$$

for $X, Y, Z, W \in \mathbb{Q}$ and since π is a p-Weil number, it must satisfy $\pi\overline{\pi} = p$, or:

$$(X^2 + dY^2 + \alpha(Z^2 + dW^2) + 2\beta dZW) + (2XY + 2\alpha ZW + \beta(Z^2 + dW^2))\sqrt{d} = p,$$

where $(\alpha, \beta) = (a, b)$, when $d \equiv 2, 3 \bmod 4$ and $(\alpha, \beta) = ((2a - b)/2, b/2)$, when $d \equiv 1 \bmod 4$. With this setting, the characteristic polynomial of Frobenius is:

$$P(x) = x^4 - 4Xx^3 + \left(2p + 4X^2 - 4dY^2\right)x^2 - 4Xpx + p^2.$$

By the first equation of System (5), the order of the Jacobian must be divisible by r. Combining the facts that p must be a prime integer, with $p \equiv \zeta_k \bmod r$ and $\#J(\mathbb{F}_p) = P(1)$, we are searching for solutions (X, Y, Z, W) of the system:

$$\left.\begin{aligned} X^2 + dY^2 + \alpha(Z^2 + dW^2) + 2\beta dZW &\equiv \zeta_k \bmod r \\ 2XY + 2\alpha ZW + \beta(Z^2 + dW^2) &= 0 \\ (\zeta_k + 1 - 2X)^2 - 4dY^2 &\equiv 0 \bmod r \end{aligned}\right\} \tag{8}$$

Remark 1. The first and third equation of System (8) are solved in $\mathbb{Z}/r\mathbb{Z}$ and the second in $\mathbb{Q}$. Such solutions are presented in [20], giving examples with $\rho \approx 8$. Alternatively, we can solve all equations modulo r and then search for lifts of X, Y, Z, W in $\mathbb{Q}$, such that the second equation is satisfied in $\mathbb{Q}$. □

Since we are working with polynomial families, we need to transfer this analysis to $\mathbb{Q}[x]/\langle r(x)\rangle$, for an $r(x) \in \mathbb{Q}[x]$ satisfying condition (2) of Definition 1 and follow Algorithm 1. We first fix a number field $L = \mathbb{Q}(\zeta_l) \cong \mathbb{Q}[x]/\langle r(x)\rangle$ for $l \in \mathbb{Z}_{>0}$, such that $k \mid l$ and set $u(x), z(x), \eta(x)$ as the polynomials representing $\zeta_l, \sqrt{d}, \eta$ in $\mathbb{Q}[x]/\langle r(x)\rangle$ (see [22,24]). We set the Frobenius polynomial:

$$\pi(x) = X(x) + Y(x) + \eta\left(Z(x) + W(x)\sqrt{d}\right), \tag{9}$$

for some $X(x), Y(x), Z(x), W(x) \in \mathbb{Q}[x]/\langle r(x)\rangle$ and the characteristic polynomial of Frobenius is now expressed in $\mathbb{Q}[t]$, with coefficients in $\mathbb{Q}[x]$. In order to construct polynomial families of pairing-friendly Jacobians we work as follows. We first solve System (8) in $\mathbb{Z}/r\mathbb{Z}$ and obtain solutions $(X, Y, Z, W) \in \mathbb{Q}^4$. Then we represent these solutions as polynomials $[X'(x), Y'(x), Z'(x), W'(x)]$ in $\mathbb{Q}[x]/\langle r(x)\rangle$ and finally we take lifts $f_X(x), f_Y(x), f_Z(x), f_W(x) \in \mathbb{Q}[x]$, so that

$$2X(x)Y(x) + 2\alpha Z(x)W(x) + \beta\left[Z(x)^2 + dW(x)^2\right] = 0,$$

namely the second equation of System (8) is satisfied in $\mathbb{Q}[x]$, where:

$$\begin{aligned} X(x) &= f_X(x)r(x) + X'(x), \quad & Y(x) &= f_Y(x)r(x) + Y'(x) \\ Z(x) &= f_Z(x)r(x) + Z'(x), \quad & W(x) &= f_W(x)r(x) + W'(x) \end{aligned}$$

The field polynomial derives from $p(x) = \pi(x)\overline{\pi}(x)$ and it must represent primes, according to Definition 1. This is equivalent to finding $m, n \in \mathbb{Z}$, such that $p(mx+n) \in \mathbb{Z}[x]$ and contains no constant or polynomial factors.

Examples of Absolutely Simple Jacobians. Let $K = \mathbb{Q}(\eta)$ be a primitive quartic CM-field and ζ_k a primitive kth root of unity. A solution of System (8) in $\mathbb{Z}/r\mathbb{Z}$ is represented by the quadruple:

$$(X, Y, Z, W) = \left(\frac{(\sqrt{\zeta_k}+1)^2}{4}, \pm\frac{(\sqrt{\zeta_k}-1)^2}{4\sqrt{d}}, \pm\frac{\zeta_k - 1}{4\eta}, \pm\frac{\zeta_k - 1}{4\eta\sqrt{d}}\right). \tag{10}$$

Below we give an example derived from the above solution, which first appeared in [8]. Our method can be also extended for arbitrary polynomials $r(x)$ satisfying condition (2) of Definition 1.

Example 1. Set $l = k = 5$ and $K = \mathbb{Q}(i\sqrt{10 + 2\sqrt{5}})$. Take $L = \mathbb{Q}(\zeta_5)$ and $r(x) = \Phi_5(x)$, so that $u(x) = x$ is a primitive 5th root of unity in $\mathbb{Q}[x]/\langle r(x)\rangle$. The representation of $\sqrt{5}$ and η in $\mathbb{Q}[x]/\langle r(x)\rangle$ is:

$$z(x) = 2x^3 + 2x^2 + 1 \qquad \text{and} \qquad \eta(x) = -2x^3 + 2x^2.$$

For $i = 4$ in Algorithm 1, and for lifts $f_X(x) = 1/4$, $f_Y(x) = 1/20$, $f_Z(x) = 1/8$ and $f_W(x) = -1/40$, we get the following solution $[X(x), Y(x), Z(x), W(x)]$:

$$\begin{aligned} X(x) &= (x^4 + 2x^2 + 1)/4, & Y(x) &= (x^4 + 6x^3 + 6x^2 + 6x + 1)/20 \\ Z(x) &= (x^4 + x^3 + 2x^2 + x + 1)/8, & W(x) &= -(x^4 + 3x^3 + 2x^2 + 3x + 1)/40 \end{aligned}$$

By Eq. (9) the Frobenius polynomial $\pi(x) \in K[x]$ is:

$$\pi(x) = X(x) + Y(x)\sqrt{5} + i\sqrt{10 + 2\sqrt{5}}\left(Z(x) + W(x)\sqrt{5}\right),$$

Setting the field polynomial as $p(x) = \pi(x)\overline{\pi}(x)$ we conclude to:

$$p(x) = \frac{1}{5}(x^8 + 2x^7 + 8x^6 + 9x^5 + 15x^4 + 9x^3 + 8x^2 + 2x + 1),$$

which is integer-valued for all $x \equiv 1 \bmod 5$. The characteristic polynomial of Frobenius $P(t)$ has integer coefficients and it is irreducible over $\mathbb{Z}$. Additionally none of conditions (2)–(5) of Proposition 1 is satisfied and the middle coefficient $B(x)$ of $P(t)$ satisfies $\gcd[B(x), p(x)] = 1$. Thus the pair $[\pi(x), r(x)]$ represents a polynomial family of pairing-friendly, absolutely simple, ordinary, 2-dimensional Jacobian varieties with embedding degree $k = 5$ and $\rho(\pi, r) = 4$. □

3.2 Generalized Dryło's Frobenius Elements

The following analysis is based on Dryło [6]. Let $K = \mathbb{Q}(\zeta_s, \sqrt{-d})$, for a square-free $d > 0$ and some primitive sth root of unity ζ_s. For quartic CM-fields K there are two cases to consider:

1. If $\sqrt{-d} \notin \mathbb{Q}(\zeta_s)$, then $\varphi(s) = 2$ and so $s \in \{3, 4, 6\}$.
2. If $\sqrt{-d} \in \mathbb{Q}(\zeta_s)$, then $\varphi(s) = 4$ and so $s \in \{5, 8, 10, 12\}$.

We take the Frobenius element $\pi \in K$ as a linear combination of ζ_s and $\sqrt{-d}$:

$$\pi = X + Y\sqrt{-d} + \zeta_s\left(Z + W\sqrt{-d}\right), \tag{11}$$

for some $X, Y, Z, W \in \mathbb{Q}$. Setting $X = Y = 0$ we recover Dryło's Frobenius elements [6] leading to non-absolutely simple Jacobian varieties. We study the case $\sqrt{-d} \notin \mathbb{Q}(\zeta_s)$ and construct the equations derived from System (4).

Let ζ_s be a primitive sth root of unity where $s \in \{3, 4, 6\}$ and so $\varphi(s) = 2$. Condition $\pi\overline{\pi} = p$ of System (4) is equivalent to:

$$\begin{aligned}\left[X^2 + Z^2 + d(Y^2 + W^2) + (\zeta_s + \overline{\zeta}_s)(XZ + dYW)\right] \\ + \left[(\zeta_s - \overline{\zeta}_s)(XW - YZ)\right]\sqrt{-d} = p\end{aligned}$$

The coefficients A, B of the characteristic polynomial of Frobenius are:

$$A = -\left[4X + 2(\zeta_s + \overline{\zeta}_s)Z\right], \quad B = 2p + (A/2)^2 + d(\zeta_s - \overline{\zeta}_s)^2 W^2$$

and so the second condition, namely $\#J(\mathbb{F}_p) \equiv 0 \bmod r$ implies:

$$[p + 1 + A/2]^2 + d(\zeta_s - \overline{\zeta}_s)^2 W^2 \equiv 0 \bmod r$$

According to the above analysis, System (5) is transformed to:

$$\left.\begin{aligned}\left[X^2+Z^2+d(Y^2+W^2)+(\zeta_s+\overline{\zeta}_s)(XZ+dYW)\right] &\equiv \zeta_k \bmod r\\ XW-YZ &= 0\\ [p+1+A/2]^2+d(\zeta_s-\overline{\zeta}_s)^2W^2 &\equiv 0 \bmod r\end{aligned}\right\} \tag{12}$$

We are working with polynomial families and so we fix the number field $L = \mathbb{Q}(\zeta_l) \cong \mathbb{Q}[x]/\langle r(x)\rangle$, where $r(x) = \Phi_l(x)$, for some $l > 0$, such that $\sqrt{d}, \zeta_s, \zeta_k \in L$. In particular this is done by setting $l = \mathrm{lcm}(s, m, k)$, where m is the smallest positive integer such that $\sqrt{d} \in \mathbb{Q}(\zeta_m)$. Then the generalized Dryło Frobenius polynomial $\pi(x) \in K[x]$ becomes:

$$\pi(x) = X(x) + Y(x)\sqrt{-d} + \zeta_s\left(Z(x) + W(x)\sqrt{-d}\right), \tag{13}$$

for some $X(x), Y(x), Z(x), W(x) \in \mathbb{Q}[x]/\langle r(x)\rangle$ and its characteristic polynomial is $P(t) \in \mathbb{Q}[t]$ as in Eq. (3), with coefficients in $\mathbb{Q}[x]$.

Examples of Absolutely Simple Jacobians with $\mathbf{s = 3}$. We give a few examples of polynomial families obtained by the solutions of System (12) for $s = 3$. Such a solution is the following:

$$\begin{aligned}X = Y &= [(\sqrt{3d}+1)(\zeta_k-1)+(\sqrt{-d}+\sqrt{-3})(\zeta_k+1)]/[2\sqrt{-3}(d+1)]\\ Z = W &= [(\zeta_k-1)+(\zeta_k+1)\sqrt{-d}]/[\sqrt{-3}(d+1)]\end{aligned} \tag{14}$$

For the second equation of System (12) there is no need to take any lifts, since Solution (14) satisfies this equation in $\mathbb{Q}$. We then expect that the constructed Jacobian varieties will have $\rho(\pi, r) < 4$.

Remark 2. In the following examples the characteristic polynomial of Frobenius $P(t)$ satisfies $P(1) \equiv 0 \bmod r(x)$, but has rational coefficients. It can be transformed to a polynomial with integer coefficients by applying a linear transformation $t \to (MT + N)$, so that for every $t \equiv N \bmod M$, we have $P(t) \in \mathbb{Z}$. □

Example 2. Let $l = 24$, so that $L = \mathbb{Q}(\zeta_{24})$. Set $r(x) = \Phi_{24}(x)$ and $u(x) = x$. For $s = 3$ and $d = 6$, the representation of $\sqrt{-6}$ and $\sqrt{-3}$ in $\mathbb{Q}[x]/\langle r(x)\rangle$ is:

$$z(x) = -2x^7 - x^5 + x^3 - x, \qquad w(x) = 2x^4 - 1,$$

respectively. For $i = 3$ in Algorithm 1 we have $k = 8$ and by Solution (14):

$$\begin{aligned}X(x) = Y(x) &= (2x^7 - 3x^6 + 3x^5 - 2x^4 - x^3 + 3x^2 + 1)/21\\ Z(x) = W(x) &= (-2x^7 - 3x^6 + 3x^5 + 2x^4 - 2x^3 - 3x - 4)/21\end{aligned}$$

The Frobenius polynomial is represented by Eq. (13), while the field polynomial is calculated by $p(x) = \pi(x)\overline{\pi}(x)$. We find that this is integer-valued for every $x \equiv \{7, 19\} \bmod 21$. It is easy to verify that none of the conditions (2)–(5) of Proposition 1 is satisfied and also $\gcd[B(x), p(x)] = 1$. Thus the pair $[\pi(x), r(x)]$ represents a family of absolutely simple, ordinary, pairing-friendly, 2-dimensional Jacobians with embedding degree $k = 8$ and $\rho(\pi, r) = 3.5$. □

Table 1. Absolutely simple Jacobians from Solution (14).

l	k	d	i	x	$\rho(\pi, r)$
24	3	6	16	$\{87, 144\} \bmod 147$	3.5000
	4		18	$\{5, 103\} \bmod 147$	
	12		2	$\{16, 94, 104\} \bmod 147$	
	24		17	$\{10, 20\} \bmod 21$	

In Table 1 we give more families derived by Solution (14). The integer $l > 0$ defined the number field $L = \mathbb{Q}(\zeta_l)$ and the 2nd column is the embedding degree, obtained by taking the ith power (4th column) of ζ_l. The 3rd column is the square-free integer $d > 0$ defining the CM-field $K = \mathbb{Q}(\zeta_3, \sqrt{-d})$. The column x refers to the congruence that the inputs of $p(x)$ must satisfy, in order to obtain integer values. Finally the last column is the ρ-value of the family. In all cases of Table 1, the characteristic polynomial of Frobenius $P(t)$ has content equal to $1/7$, which disappears by setting $t \equiv N \bmod 7$, for some $N \in \mathbb{Z}/7\mathbb{Z}$.

3.3 Alternative Representation

An alternative representation of a quartic CM-field is $K = \mathbb{Q}(\sqrt{d_1}, \sqrt{-d_2})$, for some $d_1, d_2 \in \mathbb{Z}_{>0}$, with $d_1 \neq d_2$, such that $[K : \mathbb{Q}] = 4$. Additionally, K_2 is an imaginary quadratic extension of the totally real field K_1. Then $\pi \in K$ is:

$$\pi = X + Y\sqrt{d_1} + \sqrt{-d_2}\left(Z + W\sqrt{d_1}\right), \tag{15}$$

for some $X, Y, Z, W \in \mathbb{Q}$. By the property of π being a Weil p-number, we get:

$$(X^2 + d_1Y^2 + d_2Z^2 + d_1d_2W^2) + (XY + d_2ZW)\sqrt{d_1} = p$$

and the characteristic polynomial of Frobenius is

$$P(x) = x^2 - 4Xx^3 + 4\left(X^2 - d_1Y^2\right)x^2 - 4Xpx + p^2. \tag{16}$$

Additionally, the condition $\#J(\mathbb{F}_q) = P(1) \equiv 0 \bmod r$ is equivalent to

$$(p + 1 + 2X)^2 - 4d_1Y^2 \equiv 0 \bmod r. \tag{17}$$

Using the fact that $p \equiv \zeta_k \bmod r$, we conclude to the following system:

$$\left.\begin{aligned} X^2 + d_1Y^2 + d_2Z^2 + d_1d_2W^2 &\equiv \zeta_k \bmod r \\ XY + d_2ZW &= 0 \\ (\zeta_k + 1 + 2X)^2 - 4d_1Y^2 &\equiv 0 \bmod r \end{aligned}\right\} \tag{18}$$

For polynomial families we set $L = \mathbb{Q}(\zeta_l) \cong \mathbb{Q}[x]/\langle r(x)\rangle$, where $l \in \mathbb{Z}_{>0}$ is an integer, such that $\sqrt{d_1}, \sqrt{-d_2}, \zeta_k \in \mathbb{Q}(\zeta_l)$. This is done by choosing $l = \mathrm{lcm}(m_1, m_2, k)$,

where m_1, m_2 are the smallest positive integers for which $\sqrt{d_1} \in \mathbb{Q}(\zeta_{m_1})$ and $\sqrt{-d_2} \in \mathbb{Q}(\zeta_{m_2})$. Then the Frobenius polynomial $\pi(x) \in K[x]$ is:

$$\pi(x) = X(x) + Y(x)\sqrt{d_1} + \sqrt{-d_2}\left(Z(x) + W(x)\sqrt{d_1}\right) \tag{19}$$

for $X(x), Y(x), Z(x), W(x) \in \mathbb{Q}[x]/\langle r(x)\rangle$. Note that we need to find the polynomial representation $z_1(x)$ and $z_2(x)$ of $\sqrt{d_1}$ and $\sqrt{-d_2}$ respectively in $\mathbb{Q}[x]/\langle r(x)\rangle$.

Absolutely Simple Jacobians. We give a few examples of polynomial families obtained by solving System (18). Such a solution is:

$$\begin{aligned} X &= -d_2 Z, \quad Z = \left((\zeta_k - 1) - (\zeta_k + 1)\sqrt{-d_2}\right) / \left(2(d_2 + 1)\sqrt{-d_2}\right) \\ Y &= W, \quad W = -\left((\zeta_k + 1) + (\zeta_k - 1)\sqrt{-d_2}\right) / \left(2(d_2 + 1)\sqrt{d_1}\right) \end{aligned} \tag{20}$$

For the second equation of System (18) we do not need to take any lifts, since Solution (20) satisfies this equation in $\mathbb{Q}$. Again we expect that the Jacobian families will have ρ-values less than 4. Such examples are presented in Table 2.

Remark 3. Like Remark 2, in the examples of Table 2 $P(t)$ has rational coefficients. It can be transformed into a polynomial with integer coefficients by applying a linear transformation $t \to (MT + N)$, so that for every $t \equiv N \bmod M$, we have $P(t) \in \mathbb{Z}$. An analogous transformation is also required for $p(x)$. □

Table 2. Absolutely simple Jacobians from Solution (20).

l	k	d_1	d_2	i	x	$\rho(\pi, r)$
56	7	7	2	8	$\{34, 58, 70\} \bmod 84$	3.6667
	28			2	$\{5, 47, 70\} \bmod 84$	
40	8	10	2	5	$\{5, 9, 21\} \bmod 30$	3.7500
	20			18	$\{19, 25\} \bmod 30$	

The 5th column refers to the powers i, so that $l/\gcd(l, i) = k$, while the 6th column refers to the congruence that the inputs x of the field polynomial must satisfy, in order for $p(x)$ to be an integer.

4 Implementation and Numerical Examples

The process of generating suitable Jacobian parameters when given a polynomial family $[\pi(x), r(x)]$ is summarized in Algorithm 2. This involves a simple search for some $x_0 \in \mathbb{Z}$, such that $r(x_0)$ is a large prime of a desired size. Additionally we require $p(x_0)$ to be a large prime.

In all inputs $[\pi(x), r(x)]$ of Algorithm 2 we need to ensure that $p(x)$ is integer-valued. This means that there must be integers $a, b \in \mathbb{Z}$, such that $p(x) \in \mathbb{Z}$, for all $x \equiv b \bmod a$. Algorithm 2 outputs the parameters (π, p, r). Using these

Algorithm 2. Generating suitable parameters for 2-dimensional Jacobians.

Input: A polynomial family $[\pi(x), r(x)]$ and a desired bit size S_r.
Output: A Frobenius element π, a prime q and a (nearly) prime r.

1: Find $a, b \in \mathbb{Z}$, such that $q(x) \in \mathbb{Z}$, for every $x \equiv a \mod b$.
2: Search for $x_0 \equiv b \mod a$, such that $r(x_0) = nr$, for some prime r and $n \geq 1$.
3: Set $\pi = \pi(x_0)$, $p = \pi(x_0)\overline{\pi}(x_0)$ and $r = r(x_0)/n$.
4: If $\log r \approx S_r$ and p is prime, return (π, p, r).

triples we can generate a 2-dimensional Jacobian $J(\mathbb{F}_p)$, with $r \mid \#J(\mathbb{F}_p)$ and Frobenius endomorphism π.

In all examples we considered pairing-friendly parameters of Jacobians providing a security level of at least 128 bits. These parameters are chosen according to Table 3, originally presented [1].

Table 3. Bit sizes of parameters and embedding degrees for various security levels.

Security level	Subgroup size	Extension field size	Embedding degree		
			$\rho \approx 2$	$\rho \approx 3$	$\rho \approx 4$
128	256	3000–5000	12–20	8–13	6–10
192	384	8000–10000	20–26	13–17	10–13
256	512	14000–18000	28–36	18–24	14–18

In this table we describe the sizes of the prime r, the extension field $\mathbb{F}_{p^k}$ and the ρ-values, for which we achieve a specific security level. Note that we consider only ρ-values in the range $[2, 4]$, since examples of ordinary Jacobians with $\rho < 2$ are unknown. Below we give a few numerical results.

Example 3. By Example 2 for $K = \mathbb{Q}(\zeta_3, \sqrt{-6})$, with $l = 24$ and $k = 8$:

$x_0 = 4360331437 \equiv 7 \mod 21, \quad n = 1, \quad \rho = 3.4766, \quad \log r = 256, \quad \log p = 445$

$r = 13066402029544023936014888184609183735900934642949490442553073853117438145256 1$

$p = 17104631628304699763110198214722643301043699660523612969956484320506585573386825002404876197021150163965058825820189964208554980493961 1$

The Frobenius element is given by Eq. (11), where:

$X = Y = 19977689332165391591174792446457449401947760321021273055515383733481/7$

$Z = W = -19977689345910463237518246909307679021587331482569818571002780858907/7$

Example 4. By Table 2 for $K = \mathbb{Q}(\sqrt{7}, \sqrt{-2})$, with $l = 56$ and $k = 7$:

$x_0 = 2598994 \equiv 34 \mod 84, \quad n = 1, \quad \rho = 3.6438, \quad \log r = 511, \quad \log p = 931$

$r = 902249490548244064215610498297185881520753046905674723323326873949147306603929221923916720 1638118449868190013063767741523986037176815281479499089989361$

$p = 212399046689043338177099731553386906233003858023522940013285620070426183517958274159323426053 2832768615507034012734372791904975600115792770236906989325917343177462816585017087069598844577 0188791263194455227564021970575663076557669488393474089239007369108545330451503756783697843 89$

The Frobenius element is given by Eq. (19), where:

$$X = -25696905011664630705833341687313930844434718089380678968458180995099296363752388806813678716692510902291893274011220851967761549367127 2/3$$

$$Y = 95408680352830419349196275581747456931500685432234298288318648109643042697871780410641587745583932895176493302068835260976463136917395676795/3 = W$$

$$Z = 12848452505832315352916670843656965422217359044690339484229090497549648181876194403406839358346255451145946637005610425983880774683563 6/3$$

Example 5. By Table 1 for $K = \mathbb{Q}(\zeta_3, \sqrt{-6})$, with $l = 24$ and $k = 12$:

$$x_0 = 345544178999371 \equiv 16 \bmod 147, \quad n = 1, \quad \rho = 3.4870, \quad \log r = 386, \quad \log p = 673$$

$$r = 203249108946068872403996306195052851584311711613010247013568439968339319902190248415883002212853851518240674936575281$$

$$p = 494256168316997378410237042203755744152319250884561643606630474264281418039170474898510103535450130128613678819984889519535649848076367103921678377271938018112430407319725747017112053461524001406147337 41$$

The Frobenius element is given by Eq. (14), where:

$$X = Y = 588200066152579158854583283139019284498909436781505965977463407723762779944351429654041129602047528721/7$$

$$Z = W = 588200066152578591440340377072067475914169308074137461318144114893941258301910395154905139480417211818/7$$

5 Conclusion

We presented a method for producing polynomial families of pairing-friendly Jacobians of dimension 2. We used different representations of the Frobenius element in a quartic CM-field from where we derived a system of three equations in four variables. Using the solutions of this system we constructed families of 2-dimensional, simple and ordinary Jacobians. Particularly, in this paper we focused on absolutely simple Jacobians, for which only few examples are known. The families we presented have the best ρ-values so far in the literature. We argue though that the strategy we followed in this work can be used to produce families of non-absolutely simple Jacobians as well. Finally, we provided numerical examples of suitable parameters for a security level of at least 128 bits in r-order subgroups of a Jacobian $J(\mathbb{F}_{p^k})$ and in the extension field $\mathbb{F}_{p^k}$. More examples can be derived from our proposed families by using Algorithm 2.

References

1. Balakrishnan, J., Belding, J., Chisholm, S., Eisenträger, K., Stange, K., Teske, E.: Pairings on hyperelliptic curves. In: Women in Numbers: Research Directions in Number Theory, vol. 60, pp. 87–120. Fields Institute Communications (2009). https://doi.org/10.1090/fic/060/05
2. Bernstein, D.: Elliptic vs. hyperelliptic, Part 1. Talk at ECC 2006 (2006)

3. Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Des. Codes Cryptogr. **37**(1), 133–141 (2005). https://doi.org/10.1007/s10623-004-3808-4. Springer
4. Cantor, D.G.: Computing in the Jacobian of a hyperelliptic curve. Math. Comput. **48**(177), 95–101 (1987). https://doi.org/10.1090/S0025-5718-1987-0866101-0
5. Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., Vercauteren, F.: Handbook of Elliptic and Hyperelliptic Curve Cryptography. Discrete Mathematics and its Applications. Chapman & Hall/CRC Press, Boca Raton (2006)
6. Dryło, R.: Constructing pairing-friendly genus 2 curves with split Jacobian. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 431–453. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34931-7_25
7. El Mrabet, N., Joye, M.: Guide to Pairing-Based Cryptography. CRC Press, Boca Raton (2017)
8. Freeman, D.: A generalized Brezing-Weng algorithm for constructing pairing-friendly ordinary abelian varieties. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 146–163. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85538-5_11
9. Freeman, D., Stevenhagen, P., Streng, M.: Abelian varieties with prescribed embedding degree. In: van der Poorten, A.J., Stein, A. (eds.) ANTS 2008. LNCS, vol. 5011, pp. 60–73. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79456-1_3
10. Freeman, D.M., Satoh, S.: Constructing pairing-friendly hyperelliptic curves using Weil restriction. J. Number Theor. **131**(5), 959–983 (2011). https://doi.org/10.1016/j.jnt.2010.06.003. Elsevier
11. Frey, G., Lange, T.: Fast bilinear maps from the Tate-Lichtenbaum pairing on hyperelliptic curves. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 466–479. Springer, Heidelberg (2006). https://doi.org/10.1007/11792086_33
12. Guillevic, A., Vergnaud, D.: Genus 2 hyperelliptic curve families with explicit jacobian order evaluation and pairing-friendly constructions. In: Abdalla, M., Lange, T. (eds.) Pairing 2012. LNCS, vol. 7708, pp. 234–253. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36334-4_16
13. Howe, E.W., Zhu, H.J.: On the existence of absolutely simple abelian varieties of a given dimension over an arbitrary field. J. Number Theor. **92**(1), 139–163 (2002). https://doi.org/10.1006/jnth.2001.2697. Elsevier
14. Kim, T., Jeong, J.: Extended tower number field sieve with application to finite fields of arbitrary composite extension degree. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10174, pp. 388–408. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54365-8_16
15. Kachisa, E.J.: Generating more Kawazoe-Takahashi genus 2 pairing-friendly hyperelliptic curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 312–326. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17455-1_20
16. Kawazoe, M., Takahashi, T.: Pairing-friendly hyperelliptic curves with ordinary Jacobians of type $y^2 = x^5 + ax$. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 164–177. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85538-5_12
17. Kim, T., Barbulescu, R.: Extended tower number field sieve: a new complexity for the medium prime case. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 543–571. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_20

18. Koblitz, N.: Hyperelliptic cryptosystems. J. Cryptol. **1**(3), 139–150 (1989). https://doi.org/10.1007/BF02252872. Springer
19. Lange, T.: Elliptic vs. hyperelliptic, Part 2. Talk at ECC 2006 (2006)
20. Lauter, K., Shang, N.: Generating pairing-friendly parameters for the CM construction of genus 2 curves over prime fields. Des. Codes Cryptogr. **67**(3), 341–355 (2013). https://doi.org/10.1007/s10623-012-9611-8. Springer
21. Milne, J.S.: Abelian varieties (2008). www.jmilne.org/math/
22. Murphy, A., Fitzpatrick, N.: Elliptic curves for pairing applications. CiteSeer (2005)
23. Oort, F., de Jong, A.J.: Abelian varieties over finite fields. Seminar at Columbia University, September–December 2008
24. Spearman, B.K., Williams, K.S.: Cyclic quartic fields with a unique normal integral basis. Far East J. Math. Sci. **21**, 235–240 (2006). CiteSeer
25. Weng, A.: Constructing hyperelliptic curves of genus 2 suitable for cryptography. Math. Comput. **72**(241), 435–458 (2002). https://doi.org/10.1090/S0025-5718-02-01422-9. American Mathematical Society

Statistical Testing of PRNG: Generalized Gambler's Ruin Problem

Paweł Lorek[2], Marcin Słowik[1], and Filip Zagórski[1(✉)]

[1] Department of Computer Science, Faculty of Fundamental Problems of Technology, Wrocław University of Science and Technology, Wrocław, Poland
filip.zagorski@pwr.edu.pl

[2] Faculty of Mathematics and Computer Science, Mathematical Institute, Wrocław University, Wrocław, Poland

Abstract. We present a new statistical test (*GGRTest*) which is based on the generalized gambler's ruin problem (with arbitrary winning/losing probabilities). The test is able to detect non-uniformity of the outputs generated by the pseudo-random bit generators (PRNGs).

We also propose a new method, called *BitTracker*, of processing bits of a PRNG. In most of the statistical test-suites, bits are read in 31/32-bit groups. For many tests (*e.g.,* OPERM) only a few first bits of the group are taken into account. Instead of "wasting" bits (in some statistical tests), the method takes into account every single bit of the PRNG's output.

1 Introduction

Random numbers have lots of applications, *e.g.,* in physics, simulations, gaming, gambling, cryptography (*e.g.,* for generating a secret/public key). Pseudorandom number generators (PRNGs) are the algorithms outputting numbers (bits) which *should* be indistinguishable from truly random numbers (bits). We are unable to prove that a given PRNG returns numbers which are indistinguishable from truly random ones. All we can do is to invent/design some statistical tests and to state which PRNGs fail and which do not fail them. The better the PRNG is, the more tests it should pass.

Particularly, the cryptography requires high quality PRNGs, since the output bits are used *e.g.,* to generate secret and public keys. The straightforward application of PRNGs is especially visible in the case of stream ciphers. Here the use of a weak PRNG/stream cipher may immediately lead to breaking the secrecy [2]. To evaluate the aforementioned *quality* of PRNGs, a variety of tests can be applied to their output. They aim at checking if the sequence of numbers – or bits, depending on the actual implementation – produced by PRNG resembles a sequence of elements generated independently and uniformly at random. Because of the significance of the problem, lots of testing procedures were

Authors were supported by Polish National Science Centre contract number DEC-2013/10/E/ST1/00359.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 425–437, 2017.
https://doi.org/10.1007/978-3-319-72453-9_34

developed in recent years. Most of them are based on results from the probability theory: results on bin-and-boxes schemes, results on simple random walks (*e.g.,* law of iterated logarithm [19]), duration of gamblers' ruin game [11], or hitting time distributions [10]. One can then compare the realization obtained from a PRNG with what *should* happen.

A single statistical test checks only some specific property (or few of them, in the best case) that holds for truly random numbers. If a given PRNG passes the test, it does not imply that it produces truly random numbers. It only means that the tested sequence have this particular property. That is why batteries of tests were created. They run a series of tests on output of a given PRNG. We should interpret their output as follows: the more tests are passed, the better the PRNG is. There are some famous collections of tests, *e.g.,* NIST800-90A Test Suite [4], DieHard tests (considered deprecated), DieHarder [6], or TestU01 tests [12]. The last one by L'Ecuyer and Simard is considered as a state of the art in the field.

1.1 Related Work

RC4 family. Most of the experiments were conducted on variations of RC4 algorithm, the stream cipher designed by Ronald Rivest back in 1987. RC4 and its variations are known to have defects: the weaknesses were *e.g.,* shown in [8]; distributions of some biases were shown in [9]; a famous attack on RC4 implemented in WPA-TKIP was given in [18]; weaknesses of RC4+ (which was introduced in [16]) were given [3]; in [1] authors introduced the new bias, they were able to retrieve the RC4 secret key given the state table.

Gambler's ruin problem. The tests of Kim et al. [11] (titled "Tests of randomness by the gambler's ruin algorithm") were based on the expectation and the variation of the duration of a gambler's game. Assume initially a gambler has s dollars and a bank has $N - s$ dollars. At each step the gambler either wins a dollar with probability p or loses a dollar with the remaining probability. The game ends when the gambler is broke (reaches 0) or he wins (reaches N). The expectation and variance of the duration of the game is known in specific cases (authors' algorithms $A1$–$A5$ and $B1$–$B5$, for their algorithms $C1$–$C5$ it was not known earlier). Authors split the output of a PRNG into intervals of some prespecified number of bits, each sequence is then treated as a binary representation of a number from the interval $[0, 1]$. For the starting point s the PRNG is used until 0 or N is reached, the whole procedure is repeated 20000 times and a sample of the mean of the games' duration is calculated, finally the standard Z-test is performed. Authors do not rely on a single starting point s, but for $N = 300$ they start the gambler's game for each $s = 1, \ldots, N$ and they *judge* the PRNG depending on for how many starting points it fails. They claim "hidden defects" in some generators, including commonly used Mersenne Twister generator [14]. *However*, for each starting point s the same seed is used at each iteration. Thus, the games cannot be treated as independent. For example, this is one of the obvious dependencies: if the game started at say 10 finishes at 0 (i.e., losing),

then so do the games started at $1, \ldots, 9$. This also translates in obvious way to game's duration. The authors were aware that "the Z-values corresponding to different starting points are highly correlated", nevertheless they proceeded with "we can regard a Z-value to each starting point as a separate test result". The criticism was raised by Ekkehard and Grønvik in [7], where they baldly pointed out the mistakes. Moreover, Ekkehard and Grønvik [7] show that tests of Kim et al. [11] performed *properly* do not show any defects in Mersenne Twister (and some other) PRNG.

1.2 Our Contribution

We present two main contributions.

BitTracker. Many random-walk-based tests perform the following:

S1 Split the sequence of bits returned by PRNG into sequences of pre-defined length (usually 16 or 32-bit length)
S2 Treat each one as an integer or as a number from $[0, 1]$
S3 Perform some *walk* (e.g., go *left* or *right*) according to the number obtained in S2 *independently* of the current state.
S4 Iterate S1–S3, calculate some statistics and compare them to known facts about the statistic (distribution, expectation etc.).

The main drawback of the above procedure are steps S1 and S3. Imagine gambler's ruin problem where at each step we either go left or right with probability $1/2$. Then, treating consecutive 32 bits as a binary notation of a number, we actually test every 32-nd bit. Of course if we know that the probability is $1/2$ we can simply apply 1-bit long sequence. However, such a test can overlook some dependencies between the bits. We obviate the obstacle by applying as many bits as is needed (see Sect. 3 on `BitTracker`) and by using varying probabilities (which depend on the current state). The latter feature also eliminates the drawback of S3.

Generalized gambler's ruin test. The second contribution is the new statistical test. In spirit, our approach is similar to Kim et al. [11], we also use gambler's ruin based test. However, there are two main differences:

- we consider a generalized version (with arbitrary winning/losing probabilities);
- our test is based on winning probability (instead of a game duration).

The generalization is following: being at state $i : 0 < i < N$ (for some fixed N) we go right with probability $p(i)$, go left with probability $q(i)$ and stay with probability $1 - p(i) - q(i)$. Then, for each starting point s we calculate winning probabilities using recent results from [13]. We compare it to simulations' results.

Summarizing, we will introduce test (actually, a family of tests) *GGRTest* (Generalized Gambler's Ruin Test) which heavily exploits the results on winning probability in a generalization of gambler's ruin problem (Theorem 1). There is a large number of publications introducing a single test which usually tests just one or few aspects of PRNGs. The new test proposed in this paper shows its generality – by modifying parameters it can detect different weaknesses. This is achieved due to the fact that we use varying probabilities $p(i), q(i)$ and we use "as many bits as needed" via our `BitTracker` algorithm (see Algorithm 1).

Our family of tests while it is general it still is able to spot a weak cryptographic generators like Spritz or RC4-like generators.

2 Gambler's Ruin Problem: Explicit Formulas for Winning Probabilities

Fix N and sequences $\{p(i)\}_{i=1,\ldots,N-1}, \{q(i)\}_{i=1,\ldots,N-1}$ s.t. for $i \in \{1,\ldots,N-1\}$ we have $p(i) > 0, q(i) > 0$ and $p(i) + q(i) \leq 1$. Consider Markov chain $\mathbf{X} = \{X_k\}_{k\geq 0}$ on $\mathbb{E} = \{0, 1, \ldots, N\}$ with transition probabilities

$$\mathbf{P}_X(i,j) = \begin{cases} p(i) & \text{if } j = i+1, \\ q(i) & \text{if } j = i-1, \\ 1 - (p(i) + q(i)) & \text{if } j = i, \end{cases}$$

with convention $p(0) = q(0) = p(N) = q(N) = 0$. For $i \in \{0,\ldots,N\}$ define

$$\rho(i) = P(\tau_N < \tau_0 | X_0 = i),$$

where $\tau_k = \inf\{n \geq 0 : X_n = k\}$ (note that $\rho(N) = 1$ and $\rho(0) = 0$). This is an extension of classical Gambler's ruin problem, $1 - \rho(i)$ is the ruin probability of a gambler having initially capital i. We have

Theorem 1 ([13], version simplified to 1 dimension). *Consider generalized gambler's ruin problem: for fixed N the gambler starts with capital $0 \leq s \leq N$. Having s dollars he wins 1 with probability $p(s)$ or loses 1 with probability $q(s)$ for any $\{p(i)\}_{i=1,\ldots,N-1}, \{q(i)\}_{i=1,\ldots,N-1}$ such that $p(i) > 0, q(i) > 0, p(i)+q(i) \leq 1$. Then, the probability of winning is given by*

$$\rho(s) = \frac{\sum_{n=1}^{s} \prod_{r=1}^{n-1} \left(\frac{q(r)}{p(r)}\right)}{\sum_{n=1}^{N} \prod_{r=1}^{n-1} \left(\frac{q(r)}{p(r)}\right)}. \tag{1}$$

Remark. Note that for $p(r) = p, q(r) = q$ we recover the result for classical gambler's ruin problem (with possible ties)

$$\rho(s) = \frac{\sum_{n=1}^{s} \left(\frac{q}{p}\right)^{n-1}}{\sum_{n=1}^{N} \left(\frac{q}{p}\right)^{n-1}} = \begin{cases} \frac{1-\left(\frac{q}{p}\right)^{s}}{1-\left(\frac{q}{p}\right)^{N}} & \text{if } p \neq q, \\ \frac{s}{N} & \text{if } p = q. \end{cases} \tag{2}$$

3 BitTracker: Transforming Bit Strings into Intervals

It is a common practice to interpret a string of 0s and 1s as a binary representation of fractional part of a real number from the range $[0, 1)$. Such a representation has many flaws, for instance, the significance of bits is very biased. As already mentioned, the typical approach is to split bit string into strings of equal length and then to treat each one separately as a binary representation of a number from $[0, 1)$. This is especially critical when the representation is used to place the real number in one of several intervals. For instance, given a sequence of bits '01101100' and the interval $[0, 0.1_2)$, only the first bit is actually used to determine that the real number defined by the sequence (0.01101100_2) is within the given interval.

There are also other cases, when the finite, fixed representation has other flaws, like intervals with infinite binary representations of their end points (e.g., $[\frac{1}{3}, \frac{2}{3})$).

We propose a different method of *transforming* bit string into numbers from the range $[0, 1)$. Roughly speaking, the method requires *"as many bits as needed"* to determine in which interval it is. Let $\mathcal{P} = (x_0 = 0, x_1, x_2, \ldots, x_K = 1)$ be a partition of interval [0,1). The result of the method is not a number per se, but one of the intervals. If bit string is a string of truly random and independent bits, then the probability of each interval to be selected is equal to its length. The algorithm is given in Algorithm 1.

Algorithm 1. `BitTracker`

Require: Partition $\mathcal{P} = (x_0 = 0, x_1, x_2, \ldots, x_K = 1)$, bit string $\mathcal{B} = (b_0, b_1, \ldots)$.
Ensure: Interval $[x_{i-1}, x_i) \in \mathcal{P}$ determined by $\mathcal{B}$, number of consumed bits i.

1: $i = 0, l_0 = 0, r_0 = 1$
2: **if** $\exists\ \mathcal{A} \in \mathcal{P}$ s.t. $[l_i, r_i) \subseteq \mathcal{A}$ **then**
3: **return** $\mathcal{A}, i$ and **STOP**
4: **end if**
5: take a bit b_i from the bit stream
6: split the interval $[l_i, r_i)$ into two halves $[l_i, m_i)$, $[m_i, r_i)$ where $m_i = \frac{l_i + r_i}{2}$
7: **if** $b_i = 0$ **then** proceed with the left interval: $l_{i+1} = l_i, r_{i+1} = m_i$ **else** take the right interval: $l_{i+1} = m_i, r_{i+1} = r_i$.
8: $i := i + 1$
9: **goto 2**

Sample execution of `BitTracker` for partition $\mathcal{P} = \left(0, \frac{\sqrt{2}}{2}, 1\right)$ and bit string $\mathcal{B} = 1001110$ is given in Fig. 1.

3.1 BitTracker – Every Bit Counts

The *overlapping 5-permutations test* (OPERM5) takes five consecutive random (32-bit) numbers and check if orderings of these numbers occur with statistically

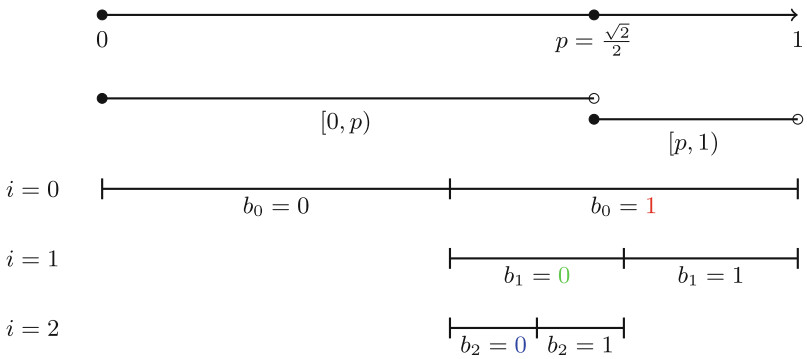


Fig. 1. `BitTracker`: Sample binary-to-interval conversion: Partition $\mathcal{P} = \left(0, \frac{\sqrt{2}}{2}, 1\right)$, bit string: $B = 1, 0, 0, 1, 1, 1, 0...$ (three first digits determine the interval)

equal probability. For the OPERM5 test only the most significant bits are taken into account. If one generates numbers in the form $x_{i,1}x_{i,2}\ldots x_{i,32}$ where $x_{i,j}$ are equal 0 and 1 with the same probability for $j = 1, \ldots, 6$ and $x_{i,j} = 0$ for $i > 6$ then the sequence of bits x_i would likely pass the test. This is because less significant bits play little role.

Now, if one applies BitTracker to the sequence and wants to apply the OPERM5 test, the situation changes.

The OPERMn takes n numbers and returns its relative order, which is then encoded into one of the possible $n!$ numbers. To perform OPERM5 with BitTracker on, one can do the following: Divide $[0, 1)$ into $[0, \frac{1}{n}), [\frac{1}{n}, \frac{2}{n}), \ldots, [\frac{n-1}{n}, 1)$ and check which interval is selected. – this is the position of the first number. To tell what is the relative position of the second number: divide $[0, 1)$ into $[0, \frac{1}{n-1}), [\frac{1}{n-1}, \frac{2}{n-1}), \ldots, [\frac{n-2}{n-1}, 1)$ and use BitTracker to tell which interval is chosen.

The position of j-th number is determined using the BitTracker on intervals: $[\frac{i}{n-j+1}, \frac{i+1}{n-j+1})$ for $i = 0, \ldots, n-j$.

4 Testing Procedure

4.1 General Approach

Fix N and sequences $\{p(i)\}_{i=1,\ldots,N-1}, \{q(i)\}_{i=1,\ldots,N-1}$ such that $p(i) > 0, q(i) > 0, p(i) + q(i) \leq 1$. From Theorem 1 we know the distribution of $Y^{(i)} = \mathbf{1}(\tau_N < \tau_0 | X_0 = i)$. Let $Y_j^{(i)}, j = 1, \ldots, n$ be *i.i.d.* samples form distribution of Y^i. The law of large numbers states that $\hat{\rho}_n(i) = \frac{1}{n}\sum_{j=1}^{n} Y_j^{(i)}$ (classical Crude Monte Carlo estimator) converges, as $n \to \infty$, to $EY^{(i)} = \rho(i)$. Moreover, from Central Limit Theorem we know that

$$Z^{(i)} = \frac{\hat{\rho}_n(i) - EY^{(i)}}{\sqrt{Var(Y^{(i)})}/\sqrt{n}} = \frac{\hat{\rho}_n(i) - \rho(i)}{\sqrt{\rho(i)(1-\rho(i))}}\sqrt{n}$$

has $N(0,1)$ distribution. $Z^{(i)}$ is often called **Z-statistics**. Thus, for a given starting point i we compare $Z^{(i)}$ with the critical value at 99% and 95%. If $|Z^{(i)}| > 2.58$ ($|Z^{(i)}| > 1.96$) we say that the point i "fails" the test at confidence level 99% (95%), *i.e.,* we mean that the numbers generated by a given PRNG and used for this point were not random. Finally, we count the number of points for which test fails and say that the PRNG fails if more than 1% (5%) of the points failed the test.

Our testing procedure $\mathsf{GGRTest}(\mathcal{G}, N, l, n, \{p(i)\}, \{\rho(s)\})$ is presented as Algorithm 2.

Algorithm 2. $\mathsf{GGRTest}(P, \mathcal{G}, N, M, l, n, \{p(i), q(i), \rho(i)\}_{i=1,\ldots,N-1})$

1: Generate $\mathcal{B} \leftarrow \mathsf{BitGeneration}(P, \mathcal{G}, M, l)$ as M l-bit long sequences from the generator $\mathcal{G}$.
2: **for all** $s \in \{1, \ldots, N-1\}$ **do**
3: Run n Gambler's ruin simulations starting at point s with the one-step probability of winning (losing) at point x is equal to $p(x)$ ($q(x)$), using bits of $\mathcal{B}$ to estimate the winning probability:

$$\hat{\rho}_n(s) = \frac{1}{n} \sum_{k=1}^{n} Y_{j,k}^{(s)},$$

where $Y_{j,k}^{(s)} = 1$ if the k-th simulation ended with winning and $Y_{j,k}^{(s)} = 0$ otherwise.
4: For obtained $\hat{\rho}_n(s)$ calculate Z-statistic:

$$Z^{(s)} = \frac{\hat{\rho}_n(s) - \rho(s)}{\sqrt{\rho(s)(1-\rho(s))}} \sqrt{n}.$$

5: Denote $\mathcal{F}_s = 1$ if $Z^{(s)} > 2.58$ and $\mathcal{F}_s = 0$ otherwise (for confidence level 99%, for confidence level 95% we set $\mathcal{F}_s = 1$ if $Z^{(s)} > 1.96$ and $\mathcal{F}_s = 0$ otherwise).
6: **end for**
7: Calculate $\mathcal{F} = \sum_{s=1}^{N-1} \mathcal{F}_s$.
8: **return** $\mathcal{F}$

4.2 Actual Parameters and Testing Procedure

We run $\mathsf{GGRTest}(P, \mathcal{G}, N, M, l, n, \{p(i)\}_{i=1,\ldots,N-1}, \{\rho(s)\}_{s=1,\ldots,N-1})$ for the following parameters:

- $N = 129$ – number of starting points,
- $M = 1$ – number of separate bit sequences,
- $l = 2^{35}$ – length of the sequence $\mathcal{B}$ of bits of a given PRNG $\mathcal{G}$ (4 GB),
- $n = 200000$ – number of simulations estimating the winning probability.
- $P =$ "GAMBLER_0001#" – per-run prefix for key derivation function.

The test was run for different generators $\mathcal{G}$ and for different gambler's ruin games. In fact we consider three different tests T1, T2, T3 which are defined

by the sequences of the winning probabilities $\{p(i)\}_{i=1,\ldots,N-1}$. In all cases we choose $p(i), q(i)$ such that $p(i) + q(i) = 1$ for all i. More precisely, the tests are defined as Algorithms 3, 4, 5.

Algorithm 3. TestT1$(P, \mathcal{G}, N, M, l, n)$

1: For $i = 1, \ldots, N-1$ set $p(i) = 1 - q(i) = 0.48$
2: Set the winning probability (the same as the winning probability of the classical Gambler's ruin problem) to:

$$\rho(s) = \frac{1 - \left(\frac{0.52}{0.48}\right)^s}{1 - \left(\frac{0.52}{0.48}\right)^N}$$

3: **return** GGRTest$(P, \mathcal{G}, N, M, l, n, \{p(i), q(i), \rho(i)\}_{i=1,\ldots,N-1})$

Algorithm 4. TestT2$(P, \mathcal{G}, N, M, l, n)$

1: For $p(i) = \frac{i}{2i+1}, q(i) = \frac{i+1}{2i+1}$,
2: Set

$$\rho(s) = \frac{\sum_{n=1}^{s} \prod_{i=1}^{n-1} \left(\frac{i+1}{i}\right)}{\sum_{n=1}^{N} \prod_{i=1}^{n-1} \left(\frac{i+1}{i}\right)} = \frac{\sum_{n=1}^{s} n}{\sum_{n=1}^{N} n} = \frac{s(s+1)}{N(N+1)}.$$

3: **return** GGRTest$(P, \mathcal{G}, N, M, l, n, \{p(i), q(i), \rho(i)\}_{i=1,\ldots,N-1})$

4.3 Bit Derivation

One more thing requires explanation. Namely, the way we "produce" bits $\mathcal{B}$ from a given PRNG $\mathcal{G}$. Recall $l = 2^{35}$ which corresponds to 4 GB.

Subsequently, the simulations are performed in the following way. Being at point s we apply `BitTracker` with partition $\mathcal{P} = (0, p(s), 1)$ (in our cases T1–T3 we have $p(s) + q(s) = 1$) with the *output* of the PRNG taken as the input. Tests reuse bits from the *output* only in case they run out of them (so-called *bit-wrapping*). We have found, that for the parameters described as above, the sequences wrap at most twice.

5 Experimental Results

We applied the procedure described in Sect. 4 to the following PRNGs:

- **RC4.** The above described RC4 with key of size 256 bits
- **RC4-64.** The version of RC4 using key of size 64 bits, where first 24 bits correspond to an IV.

- **RC4+** was introduced in 2008 in [16]. It is a modification of RC4 that introduced much more complex three-phase key schedule algorithm (taking about 3× as long as RC4) and slightly modified generation function.
- **RC4A** was introduced in 2004 in [17]. It is a modification of RC4 that is based on two independent internal permutations (states) interleaved in generation process.
- **VMPC** was introduced in 2004 in [20]. It is a modification of RC4 that uses a slightly modified KSA and a PRNG.
- **Spritz** is a recent (2014) sponge construction by Ronald Rivest that bears constructional similarities to RC4, see [15] for details. It can be used as random bit generator
- **Salsa20** which was introduced in 2008 in [5]. The stream cipher is based on pseudorandom function and ARX (add-rotate-xor) operations.
- **AES256-CTR.** AES with 256-bit key, counter mode.
- **Urandom.**
- **RandU LCG** known to be totally broken.

Algorithm 5. TestT3$(P, \mathcal{G}, N, M, l, n)$

1: For $i = 1, \ldots, N-1$ set $p(i) = \frac{i^3}{i^3+(i+1)^3}, q(i) = \frac{(i+1)^3}{i^3+(i+1)^3}$.
2: Set

$$\rho(s) = \frac{\sum_{n=1}^{s} n^3}{\sum_{n=1}^{N} n^3} = \frac{s^2(s+1)^2}{N^2(N+1)^2}.$$

3: **return** GGRTest$(P, \mathcal{G}, N, M, l, n, \{p(i), q(i), \rho(i)\}_{i=1,\ldots,N-1})$

Algorithm 6. BitGeneration$(P, \mathcal{G}, M, l)$

1: **for all** r in range $\{1, \ldots, M\}$ **do**
2: Calculate master key $K_r =$ SHA-256$(P||r)$
3: The key used for $\mathcal{G}$ is computed as λ-byte suffix of K_r, where λ denotes the length of the PRNG key.
3: Run $\mathcal{G}$ to obtain an l-bit long string $\mathcal{B}[r] = \mathcal{G}(K_r)_{|l}$.
4: **end for**
5: **return** $\mathcal{B}$

In Fig. 2 we present results of one experiment: number of points failing test means number of starting points for which the (absolute value of) z-statistic was larger then 2.58 (for confidence level 99%) or 1.96 (for confidence level 95%). Note that for 128 starting points we should have, on average 1.28 (for conf. level 99%) or 6.4 (for conf. level 95%) failing points.

The z-statistics for chosen PRNGs are presented in Fig. 3. The Z-statistics for RandU deviated significantly from the others, therefore we placed them in separate plot.

	Confidence Level 99%			Confidence level 95%		
	T1	T2	T3	T1	T2	T3
AES256CTR	2	3	3	8	10	10
RandU	115	123	110	119	123	112
RC4	1	1	3	7	3	6
RC4-64	0	2	2	4	7	10
RC4a	1	1	4	7	10	6
RC4+	0	1	1	5	3	4
Salsa20	2	1	1	5	6	8
Spritz	3	2	3	7	10	7
urandom	1	2	2	9	7	8
VMPC	0	2	1	0	8	3

Fig. 2. Simulations results for $N = 129$ and T1–T3. Number of points for which given test failed.

5.1 The χ^2 Test

We performed 31 times the experiment (whose results are presented in Fig. 2) for $T2$ and starting points 41–122 (we also excluded randu). Let O_i denotes number of times (out of 31) for the test failed for i starting points. Note that each point fails with probability 0.05 (conf. level 95%) or 0.01 (conf. level 99%), thus the number of failing points has $Bin(82, p)$ distribution ($p = 0.05$ for conf. level 95% and $p = 0.01$ for conf. level 99%). Thus, on average, there should be $E_i = 31\binom{82}{i}p^i(1-p)^{81-i}$ failing points observed. We can test it using $\tilde{\chi}^2$ statistics:

$$\tilde{\chi}^2 = \sum_{i=1}^{n_0} \frac{(O_i - E_i)^2}{E_i}.$$

In our experiments the maximal observed value was 12 (*i.e.,* in one of the simulations there were 12 (for conf. level 95%) failing points (6 for conf. level 99%) for some specific - Spritz - test), we truncate it to $n_0 = 13$ (conf. level 95%) or $n_0 = 7$ (conf. level 99%). Thus, the $\tilde{\chi}^2$ statistic has χ^2 distribution with 12 (or 6) degrees of freedom. The results are presented in Fig. 4. For χ^2 Test we excluded RandU, urandom and included Mersenne-Twister.

6 Summary

We presented a new, general technique called *BitTracker* – to handle bit sequences that are about to feed a statistical test. We also proposed and implemented a family of statistical tests which are based on properties of the generalized gambler's ruin problem. The preliminary results show that the new test-family has a potential to spot weaknesses in cryptographic bit generators (*e.g.,* Spritz).

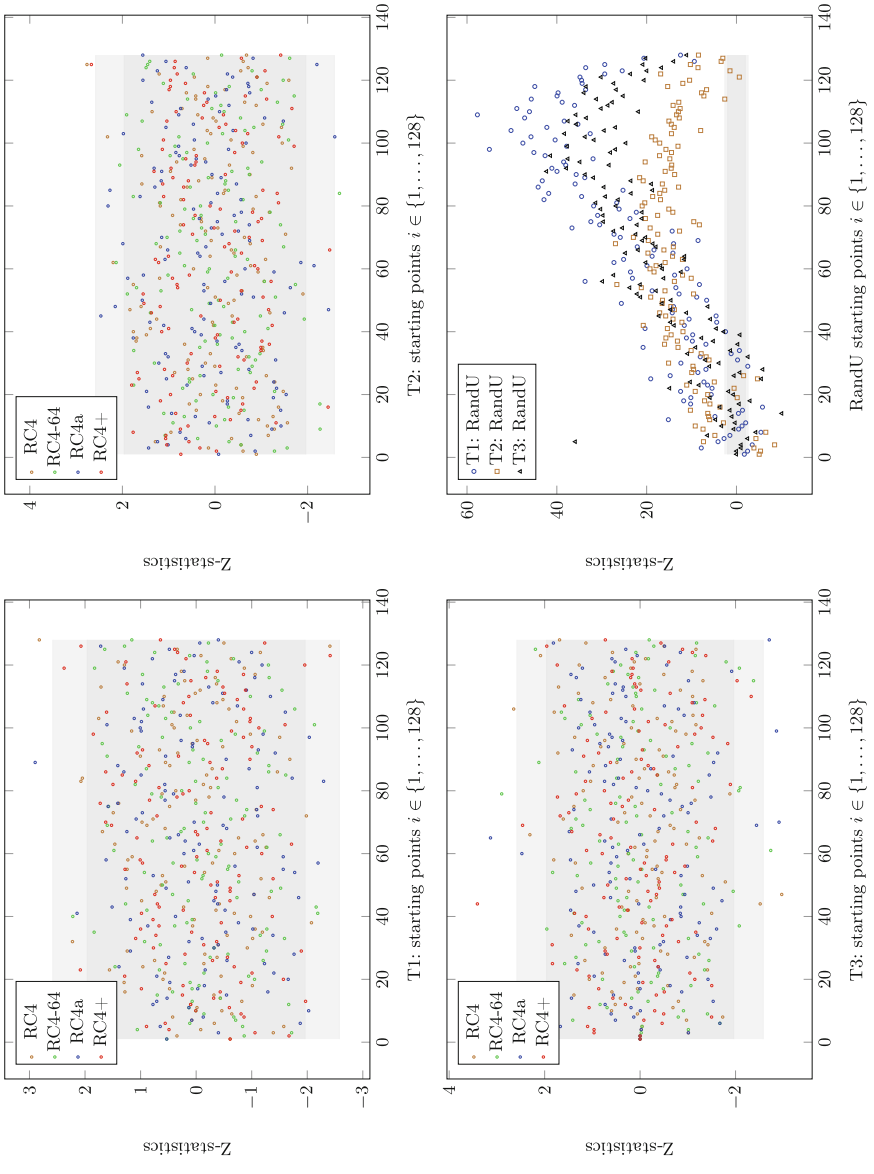


Fig. 3. Z-statistics obtained from simulations for tests T1–T3: RC4, RC4-64, RC4a, RC4+. RandU presented separately on last plot.

	Confidence Level 99%		Confidence level 95%	
	$\tilde{\chi}^2$	p-value	$\tilde{\chi}^2$	p-value
AES256CTR	5.6305	0.4658	23.1988	0.0260
Mersenne Twister	9.6979	0.13796	43.6676	1.7395e-05
RC4-64	30.9427	2.5994e-05	14.9915	0.24190
RC4	3.3817	0.75963	10.2043	0.5980
RC4a	9.6979	0.13796	21.5660	0.0426
RC4+	16.1583	0.0129	27.9784	0.0055
Salsa20	6.2032	0.4008	43.9593	1.5516e-05
Spritz	34.7307	4.8593e-06	70.1628	2.9859e-10
VMPC	31.8370	1.7533e-05	27.0109	0.0076

Fig. 4. χ^2 test for 31 experiments for $T2$, starting points 41–122.

References

1. Akgün, M., Kavak, P., Demirci, H.: New results on the key scheduling algorithm of RC4. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 40–52. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89754-5_4
2. AlFardan, N., Bernstein, D.J., Paterson, K.G., Poettering, B., Schuldt, J.C.N.: On the security of RC4 in TLS. In: Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13), Washington, D.C., pp. 305–320. USENIX (2013)
3. Banik, S., Sarkar, S., Kacker, R.: Security analysis of the RC4+ stream cipher. In: Paul, G., Vaudenay, S. (eds.) INDOCRYPT 2013. LNCS, vol. 8250, pp. 297–307. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03515-4_20
4. Barker, E., Kelsey, J.: DRAFT NIST Special Publication 800-90A, Rev. 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators. Technical report, NIST (2014)
5. Bernstein, D.J.: The Salsa20 family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 84–97. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_8
6. Brown, R.G., Eddelbuettel, D., Bauer, D.: Dieharder: a random number test suite. www.phy.duke.edu/~rgb/General/dieharder.php
7. Ekkehard, H., Grønvik, A.: Re-seeding invalidates tests of random number generators. Appl. Math. Comput. **217**(1), 339–346 (2010)
8. Fluhrer, S., Mantin, I., Shamir, A.: Weaknesses in the key scheduling algorithm of RC4. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 1–24. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45537-X_1
9. Jha, S., Banik, S., Isobe, T., Ohigashi, T.: Some proofs of joint distributions of keystream biases in RC4. In: Dunkelman, O., Sanadhya, S.K. (eds.) INDOCRYPT 2016. LNCS, vol. 10095, pp. 305–321. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49890-4_17
10. Kang, M.: Efficiency test of pseudorandom number generators using random walks. J. Comput. Appl. Math. **174**(1), 165–177 (2005)
11. Kim, C., Choe, G.H., Kim, D.H.: Tests of randomness by the gambler's ruin algorithm. Appl. Math. Comput. **199**(1), 195–210 (2008)

12. L'Ecuyer, P., Simard, R.: TestU01: a C library for empirical testing of random number generators. ACM Trans. Math. Softw. **33**(4), 22-es (2007)
13. Lorek, P.: Generalized gambler's ruin problem: explicit formulas via Siegmund duality. Methodol. Comput. Appl. Prob. **19**(2), 603–613 (2017)
14. Matsumoto, M., Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8**(1), 3–30 (1998)
15. Schuldt, J.C.N., Rivest, R.L.: Spritz—a spongy RC4-like stream cipher and hash function. Technical report (2014)
16. Paul, S., Preneel, B.: A new weakness in the RC4 keystream generator and an approach to improve the security of the cipher. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 245–259. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-25937-4_16
17. Maitra, S., Paul, G.: Analysis of RC4 and proposal of additional layers for better security margin. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 27–39. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89754-5_3
18. Vanhoef, M., Piessens, F.: All your biases belong to us: breaking RC4 in WPA-TKIP and TLS. In: USENIX Security Symposium (2015)
19. Wang, Y., Nicol, T.: On statistical distance based testing of pseudo random sequences and experiments with PHP and Debian OpenSSL. Comput. Secur. **53**, 44–64 (2015)
20. Zoltak, B.: VMPC one-way function and stream cipher. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 210–225. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-25937-4_14

Subtleties in Security Definitions for Predicate Encryption with Public Index

Johannes Blömer(✉) and Gennadij Liske

Paderborn University, Paderborn, Germany
{bloemer,gennadij.liske}@upb.de

Abstract. We take a critical look at established security definitions for predicate encryption (PE) with public index under chosen-plaintext attack (CPA) and under chosen-ciphertext attack (CCA). We identify three different formalizations of key handling in the literature, implicitly assumed to lead to the same security notion. Contrary to this assumption, we prove that the corresponding models result in two different security notions under CPA and three different security notions under CCA. Similarly to the recent results for PKE and conventional key-encapsulation mechanism (KEM), we analyze subtleties in the security definitions for PE and predicate key-encapsulation mechanism (P-KEM) regarding the so-called "no-challenge-decryption" condition. While the results for PE and PKE are similar, the results for P-KEM significantly differ from the corresponding results for conventional KEM. As a consequence of our work, we suggest security definitions for PE and P-KEM under different attack scenarios.

Keywords: Predicate encryption with public index
Key-encapsulation mechanism · Chosen-plaintext attack
Chosen-ciphertext attack

1 Introduction

Over the last decades, cryptographic primitives and schemes considered in modern cryptography have become more complex. Therefore, it is important that security models are rigorously studied and made accessible to the cryptographic community through comprehensive and perspicuous explanations, especially when these models are translated into new contexts. In a recent work Bellare et al. [5] have analyzed security definitions for public-key encryption under chosen-ciphertext attacks, which were assumed to lead to the same security notion. They proved that these definitions are not all equivalent and that some of them are too weak. If even these security definitions suffer from weaknesses, what about new and more involved cryptographic schemes and their security models, which are often justified by their origin in *well-known* definitions?

The authors were partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901).

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 438–453, 2017.
https://doi.org/10.1007/978-3-319-72453-9_35

In this paper we look at predicate encryption (PE) with public index and predicate key-encapsulation mechanism (P-KEM) with public index. On the one hand, these cryptographic primitives have been extensively studied due to their usefulness in cryptographic applications. On the other hand, established security definitions for PE and P-KEM lean to a large extent on the corresponding security definitions for PKE and KEM, even though the functionality of predicate-based schemes is more complex compared to conventional PKE and KEM.

PE with public index (also payload hiding PE) is a relatively new but already established and well-studied primitive. It can be used to realize fine-grained access control to data by cryptographic encryption. In a PE scheme for predicate R the data are encrypted under ciphertext indices cInd, which specify access requirements and are not confidential. The users hold secret keys with key indices kInd, which represent their access rights. A user with a secret key for kInd can reconstruct the message, encrypted under cInd, if the predicate is satisfied by the indices, that is if $\mathrm{R}(\mathrm{kInd}, \mathrm{cInd}) = 1$. In turn, P-KEM can be seen as specialized PE, which is used to encrypt a random bit string. P-KEMs are applied in hybrid constructions together with symmetric-key encryption schemes.

The study of PE started when Shamir introduced the idea of identity-based encryption (IBE) [23], a predicate encryption for the equality predicate. The first fully-functional IBE was presented in [10]. The study of PE for more sophisticated predicates started with [22]. Furthermore, the more general concept of functional encryption (FE) was introduced in [12].

If we look at the security models previously used in the context of PE, we recognize that they originate from the security models for IBE (cf. [22]), which in turn go back to the security models for PKE (cf. [10]). Starting with [10] the indistinguishability (IND) definition for PKE was adapted and used for IBE [16–18,24] and for more sophisticated PE [1,3,7,19,20,22,25,26]. Consequently, in [2] security models for IBE were studied under different attack scenarios. One of the results of this work was an equivalence proof of semantic security definitions and IND-definitions. In contrast, for the more general context of functional encryption the IND-definition was proved not to be suitable (cf. [12,21]). Surprisingly, and seemingly contradicting the previous results, in [12] the authors also proved that semantic security can not be achieved even for IBE. This impossibility result was identified in [4,6] as a consequence of the so-called key-revealing selective-opening attacks (SOA-Ks), which were implicitly covered by the semantic security definitions of [12,21], but were not considered in [2]. The analysis of [4,6,12,21] was restricted to the CPA attack scenario.

Main Contributions. In this paper we use indistinguishability of encryption under CCA as our basic security model. For a proof that semantic security and indistinguishability of encryption under CCA are equivalent for PE as long as the SOA-Ks are not considered, we refer to the full version of this work [8]. This equivalence result is not surprising, but closes the gap between [2] and [4,6,12,21]

How to handle user secret keys? Whereas in the context of conventional PKE there is only a single secret key in question, in PE schemes there are many user

secret keys generated from the master secret key by a trusted authority. Hence, the authority manages access to the encrypted data. Several users may hold different keys for the same key index. Already security definitions for IBE [10] explicitly prevent user collusion and formalize this by an additional key generation oracle. We identify three different formalizations for PE regarding the user secret keys in the literature and name these as follows:

`One-Key model` (OK-model)
`One-Use model` (OU-model)
`Covered-Key model` (CK-model).

In the first two models the adversary (denoted by $\mathcal{A}$) has access to the oracles $\mathbf{KGen}(\cdot)$ and $\mathbf{Dec}(\cdot,\cdot)$, when the CCA scenario is considered. For the first oracle $\mathcal{A}$ specifies a key index and receives a secret key for this key index. For the second oracle $\mathcal{A}$ specifies a ciphertext as well as a key index, and the ciphertext is decrypted using a secret key for the specified key index. The OK-model and the OU-model differ in the handling of the user secret keys in these oracles. In the OK-model a unique secret key for kInd is generated and stored if this index is submitted by $\mathcal{A}$ for the first time. This user secret key is used to answer all oracle queries related to kInd. In turn, in the OU-model the challenger generates a new secret key for every key generation query and for every decryption query. Hence, every generated user secret key is used only once. Under CCA the OK-model has previously been used e.g. in [10,16,25,26] and the OU-model has previously been used e.g. in [11,17]. The CK-model has previously been used in [7]. In this model the user secret keys are generated and numbered in the co-called covered key generation oracle $\mathbf{CKGen}(\cdot)$. The adversary can then ask to reveal the generated keys. For decryption queries $\mathcal{A}$ specifies the number of the secret key which has to be used for decryption. This models the fact that different keys, even for the same key index, are held and used by different users and an adversary might be able to realize chosen-ciphertext attacks on these users and their secret keys (cf. [7]).

The three identified models have previously been used and most researcher seem to think they refer to the same security notion. However, already in [9] Boneh et al. shortly discuss the OK-model and the OU-model for the case that the key generation algorithm in IBE is probabilistic. They state that "*The resulting security definitions [...] seem incomparable, and there does not appear to be a reason to prefer one over the other.*" For PE we prove that under CCA the three security notions are different and that the security notion achieved from the CK-model is the strongest one. The weakness of the OK-model is not surprising, since already under CPA there are OK-secure IBE schemes which are not OU-secure (cf. [11,13]). The weakness of the OU-model and its relation to the CK-model under CCA is more surprising. Except for [7], the CK-model has not been considered and hence, we examine the CK-security of known OK/OU-secure PE schemes.

When and how should a challenge decryption be disallowed? We consider this question in the context of PE and P-KEM following the results in [5] for PKE

and KEM. While it is not surprising that we can prove similar results for PE, the situation is different when KEMs are considered. Namely, in the context of conventional KEM six different security notions were identified and proved to be equivalent in [5]. First of all, we consider two additional security notions (due to the additional key generation oracle) and prove that four of the eight security definitions are too weak in general. The other four notions are in fact equivalent, although some reductions between these notions are not tight. As a consequence of our results, in the context of P-KEM one can disallow the decapsulation query on the challenge encapsulation only in the second query phase and can, equivalently, model this restriction both in the penalty-style (SP) and in the exclusion-style (SE). In contrast to this result, the first query phase can be completely dropped for conventional KEM [5].

Conclusion. Summarizing our results, under chosen-ciphertext attack we suggest to use the CK-model to handle user secret keys. Under chosen-plaintext attack the simpler OU-model is appropriate. This suggestion holds for PE as well as for P-KEM. Finally, under CCA the SE-model is the most advisable model to handle the no-challenge-decryption condition for both PE and P-KEM.

Organization. In Sect. 2 we present definitions of PE and P-KEM. Section 3 contains indistinguishability templates for security definitions of PE and P-KEM. Based on these templates we look at different formalizations regarding handling of user secret keys in Sect. 4. Finally, in Sect. 5 we consider security notions originating from restrictions of adversaries to query the decryption of the challenge ciphertext.

2 Preliminaries

We denote by $\alpha := a$ the assignment of the value a to the variable α. Let X be a random variable on a finite set S. We denote by $[X]$ the support of X. This notation can be extended to probabilistic polynomial time (ppt) algorithms, since every ppt algorithm $\mathcal{A}$ on input x defines a finite output probability space which we denote by $\mathcal{A}(x)$. That is, $[A(x)]$ denotes the set of all possible outcomes of A on input x. We write $\alpha \leftarrow X$ to denote the sampling of an element from S according to the distribution defined by X ($y \leftarrow A(x)$ for ppt algorithms). We also write $\alpha \leftarrow S$ when sampling an element from S according to the uniform distribution.

2.1 Predicate-Based Schemes with Public Index

Let Ω, Σ be arbitrary sets. A *predicate family* $\mathcal{R}_{\Omega,\Sigma}$ is a set of binary relations $\mathcal{R}_{\Omega,\Sigma} = \left\{\mathrm{R}_\kappa : \mathbb{X}_\kappa \times \mathbb{Y}_\kappa \to \{0,1\}\right\}_{\kappa\in\Omega\times\Sigma}$, where $\mathbb{X}_\kappa$ and $\mathbb{Y}_\kappa$ are sets called the *key index space* and the *ciphertext index space* of R_κ, respectively. The following conditions must hold:

– *Membership test for* $\mathbb{X}_\kappa$: There exists a polynomial-time algorithm which on input $(\kappa, \mathrm{kInd}) \in (\Omega \times \Sigma) \times \{0,1\}^*$ returns one if and only if $\mathrm{kInd} \in \mathbb{X}_\kappa$.

- *Membership test for* $\mathbb{Y}_\kappa$: There exists a polynomial-time algorithm which on input $(\kappa, \text{cInd}) \in (\Omega \times \Sigma) \times \{0,1\}^*$ returns one if and only if $\text{cInd} \in \mathbb{Y}_\kappa$.
- *Easy to evaluate*: There exists a polynomial-time algorithm which on input $(\kappa, \text{kInd}, \text{cInd}) \in (\Omega \times \Sigma) \times \mathbb{X}_\kappa \times \mathbb{Y}_\kappa$ returns $\mathrm{R}_\kappa(\text{kInd}, \text{cInd})$.

For example, in key-policy ABE $\mathbb{X}_\kappa$ is a set of Boolean formulas ϕ over variables $x_1, \ldots, x_n$, set $\mathbb{Y}_\kappa$ is the power set of the x_i's, and $\mathrm{R}_\kappa(\phi, \gamma) = 1 \Leftrightarrow \phi(\gamma) = 1$ ($x_i \in \gamma$ for $\gamma \in \mathbb{Y}_\kappa$ means $x_i = 1$). Let $\kappa = (\text{des}, \sigma)$. Indices $\text{des} \in \Omega$ specify some general description properties of the corresponding predicates (e.g. the size of γ in the example above might be restricted), and indices $\sigma \in \Sigma$ specify the domains of computation which depend on the security parameter (e.g. $\mathbb{Z}_p$).

Definition 2.1. *A* predicate key encapsulation mechanism with public index Π *for predicate family* $\mathcal{R}_{\Omega,\Sigma}$ *and family of key spaces* $\mathcal{K} = \{\mathbb{K}_\lambda\}_{\lambda \in \mathbb{N}}$ *consists of four ppt algorithms:*

Setup $\left(1^\lambda, \text{des}\right) \to (\text{msk}, \text{pp}_\kappa)$**:** *Setup algorithm generates a master secret key* msk *and public parameters* pp_κ *for index* $\kappa = (\text{des}, \sigma) \in \Omega \times \Sigma$.

KeyGen $\left(1^\lambda, \text{pp}_\kappa, \text{msk}, \text{kInd}\right) \to \text{sk}$**:** *Key generation algorithm outputs a user secret key* sk *for key index* kInd.

Encaps $\left(1^\lambda, \text{pp}_\kappa, \text{cInd}\right) \to (\text{K}, \text{CT})$**:** *Encapsulation algorithm outputs a key* $\text{K} \in \mathbb{K}_\lambda$ *and an encapsulation* CT *of* K *under ciphertext index* cInd.

Decaps $\left(1^\lambda, \text{pp}_\kappa, \text{sk}, \text{CT}\right) \to \text{K}$**:** *Decapsulation algorithm outputs a key* $\text{K} \in \mathbb{K}_\lambda$ *or the error symbol* $\perp \notin \mathbb{K}_\lambda$.

Correctness: *For every security parameter* λ*, every* $\text{des} \in \Omega$*, every* $(\text{msk}, \text{pp}_\kappa) \in \left[\text{Setup}\left(1^\lambda, \text{des}\right)\right]$*, every* $\text{kInd} \in \mathbb{X}_\kappa$*,* $\text{cInd} \in \mathbb{Y}_\kappa$ *which satisfy* $\mathrm{R}_\kappa(\text{kInd}, \text{cInd}) = 1$*, and every* $(\text{K}, \text{CT}) \in \left[\text{Encaps}\left(1^\lambda, \text{pp}_\kappa, \text{cInd}\right)\right]$ *it must hold*

$$\Pr\left[\text{Decaps}\left(1^\lambda, \text{pp}_\kappa, \text{KeyGen}\left(1^\lambda, \text{pp}_\kappa, \text{msk}, \text{kInd}\right), \text{CT}\right) = \text{K}\right] = 1.$$

We assume for convenience that the public parameters pp_κ have length at least λ, and that λ and $\kappa \in \Omega \times \Sigma$ can be efficiently determined from pp_κ. Hence, we avoid to write 1^λ as input of the algorithms except for the setup algorithm. Furthermore, if the public parameters are fixed and obvious from the context, we also avoid to write pp_κ as input of the algorithms. The definition of predicate encryption for message space $\mathcal{M}$ is as usual. The algorithms Encaps and Decaps are replaced by algorithms Enc and Dec, respectively.

3 Indistinguishability Templates for PE and P-KEM

Next, we define the indistinguishability templates for PE as well as for P-KEM. These templates are based on the usual formalization of indistinguishabiity experiments for encryption schemes (see for example [15]). In these templates we abstract from concrete attack scenarios, denoted by ATK. Hence, the oracles remain unspecified for the time being. Furthermore, for later purposes we use a

$\mathbf{IND\text{-}PE}^{\mathrm{ATK}}_{\Pi,\mathcal{A}}(\lambda, \mathrm{des})$:

$b \leftarrow \{0,1\}$; $(\mathrm{msk}, \mathrm{pp}_\kappa) \leftarrow \mathrm{Setup}\left(1^\lambda, \mathrm{des}\right)$;

$(\mathrm{cInd}^*, m_0, m_1, St) \leftarrow \mathcal{A}_1^{\mathbf{O}_1^{\mathrm{ATK}}(\cdot)}\left(1^\lambda, \mathrm{pp}_\kappa\right)$;

Output $b \wedge \overline{\mathrm{BadQuery}}$ if $\mathcal{A}_1$ outputs $\perp$;

$\mathrm{CT}^* \leftarrow \mathrm{Enc}\left(\mathrm{pp}_\kappa, \mathrm{cInd}^*, m_b\right)$;

$b' \leftarrow \mathcal{A}_2^{\mathbf{O}_2^{\mathrm{ATK}}(\cdot)}\left(\mathrm{CT}^*, St\right)$;

Output $b = b' \wedge \overline{\mathrm{BadQuery}}$;

$\mathbf{P\text{-}KEM}^{\mathrm{ATK}}_{\Pi,\mathcal{A}}(\lambda, \mathrm{des})$:

$b \leftarrow \{0,1\}$; $(\mathrm{msk}, \mathrm{pp}_\kappa) \leftarrow \mathrm{Setup}\left(1^\lambda, \mathrm{des}\right)$;

$(\mathrm{cInd}^*, St) \leftarrow \mathcal{A}_1^{\mathbf{O}_1^{\mathrm{ATK}}(\cdot)}\left(1^\lambda, \mathrm{pp}_\kappa\right)$;

Output $b \wedge \overline{\mathrm{BadQuery}}$ if $\mathcal{A}_1$ outputs $\perp$;

$(\mathrm{K}_0, \mathrm{CT}^*) \leftarrow \mathrm{Encaps}\left(\mathrm{pp}_\kappa, \mathrm{cInd}^*\right)$; $\mathrm{K}_1 \leftarrow \mathbb{K}_\lambda$;

$\mathrm{K}^* := \mathrm{K}_b$;

$b' \leftarrow \mathcal{A}_2^{\mathbf{O}_2^{\mathrm{ATK}}(\cdot)}\left(\mathrm{K}^*, \mathrm{CT}^*, St\right)$;

Output $b = b' \wedge \overline{\mathrm{BadQuery}}$.

Fig. 1. Indistinguishability experiments for PE and P-KEM.

so-called BadQuery event in the experiments, which can be utilized to specify restrictions of adversarial queries. The convention is that the BadQuery event occurs if an adversary violates these restrictions.

Let Π be a PE scheme for predicate family $\mathcal{R}_{\Omega,\Sigma}$ and message space $\mathcal{M}$. An indistinguishability adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against Π is a pair of algorithms with oracle access such that $\mathcal{A}_1$, given correctly generated public parameters pp_κ, outputs the error symbol $\perp$ or a tuple $(\mathrm{cInd}^*, m_0, m_1, St)$ satisfying $\mathrm{cInd}^* \in \mathbb{Y}_\kappa$, $m_0, m_1 \in \mathcal{M}$ and $|m_0| = |m_1|$. Algorithm $\mathcal{A}_2$ always outputs a bit. The ciphertext index cInd^* is called the challenge ciphertext index. The set of IND-adversaries against Π is denoted by $\mathbf{A}^{\mathrm{IND}}_{\Pi,\mathrm{PE}}$ or just by $\mathbf{A}^{\mathrm{IND}}_{\mathrm{PE}}$, if Π is obvious from the context. The set $\mathbf{A}^{\mathrm{IND}}_{\Pi,\text{P-KEM}}$ of adversaries against P-KEM scheme Π is defined similarly except for the output of $\mathcal{A}_1$, which does not contain messages. The indistinguishability experiments are presented in Fig. 1, where in the case of P-KEM the family of key spaces of Π is denoted by $\mathcal{K} = \{\mathbb{K}_\lambda\}_{\lambda\in\mathbb{N}}$.

Differently from the usual indistinguishability experiments for PE, we allow A_1 to abort using the error symbol $\perp$. We explain this modification next. In security definitions for PE the adversary is allowed to query secret keys, which models collusion attacks. We call a key index kInd a corrupted key index, if $\mathcal{A}$ queries a key for kInd. $\mathcal{A}$ is prohibited to corrupt kInd if $\mathrm{R}(\mathrm{kInd}, \mathrm{cInd}^*) = 1$, where cInd^* is the challenge ciphertext index. To the best of our knowledge, in all previous security definitions for PE this restriction has always been modeled in exclusion-style (in terms of [5]). That is, adversaries which violate this restriction are not considered at all. Due to our modification, we can formally justify this restriction for the first query phase. Furthermore, we remark that $\mathcal{A}$ cannot increase her advantage (defined in the following section) using the error symbol $\perp$ as output after the first query phase, but this behavior is also not penalized. Indeed, as long as $\mathcal{A}_1$ does not cause the event BadQuery, the output of the experiment is 1 with probability $1/2$ in the case when $\mathcal{A}_1$ outputs $\perp$.

4 Handling of User Secret Keys

Whereas in the context of conventional PKE there is only a single secret key in question, in predicate-based schemes there are many user secret keys generated

Table 1. Oracle specification for different models under CCA attacks.

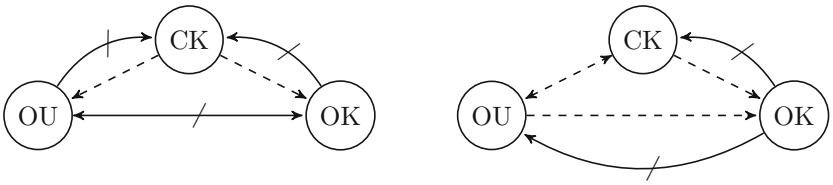
OK	OU	CK
KGen (kInd): If (kInd, sk) $\in S_k$ return sk; sk $\leftarrow$ KeyGen (msk, kInd); S_k.add ((kInd, sk)); Return sk; **Decaps** (CT, kInd): sk := **KGen** (kInd) Return Decaps (sk, CT);	**KGen** (kInd): sk $\leftarrow$ KeyGen (msk, kInd); Return sk; **Decaps** (CT, kInd): sk := **KGen** (kInd) Return Decaps (sk, CT);	**CKGen** (kInd): sk $\leftarrow$ KeyGen (msk, kInd); i++; S_k.add ((i, sk)); **Open** (i): Return sk from $(i, \text{sk}) \in S_k$; **Decaps** (CT, i): Return Decaps (sk, CT), if $(i, \text{sk}) \in S_k$;

S_k contains all keys which have been generated by **KGen** / **CKGen**.

from the same master secret key. Actually, several users may hold different keys for the same key index. The goal of this section is to consider different ways of handling user secret keys in security experiments. We identify three different formalizations in the literature and name these as follows: one-key model (OK-model), one-use model (OU-model), and covered key model (CK-model). The oracles for these models under CCA attacks are presented in Table 1. As mentioned in the previous section we can assume w.l.o.g. that $\mathcal{A}$ never corrupts a key index kInd if $\mathrm{R}(\text{kInd}, \text{cInd}^*) = 1$. Hence, in this section we use the BadQuery event to penalize $\mathcal{A}_2$ for a decapsulation query on CT^* if $\mathrm{R}(\text{kInd}, \text{cInd}^*) = 1$ for the challenge ciphertext index cInd^* and the key index kInd specified for this decapsulation query. Under CPA attacks the decapsulation oracle is dropped.

In the OK-model the challenger generates and stores a unique secret key for kInd, if this index is submitted by $\mathcal{A}$ for the first time. This user secret key is used to answer all oracle queries related to kInd. In particular, the oracle query **KGen**(kInd) always results in the same key. The OK-model was previously used e.g. in [10,16,25,26]. In the OU-model the challenger generates a new secret key for every query and the generated key is used only once. This model was previously used e.g. in [11,17]. In the CK-model the adversary specifies not only the key indices, but also the keys which are used to answer the decryption queries. This is formalized using an additional covered key generation oracle. The CK-model reflects the fact that users hold specific secret keys and use their keys for decryption. Hence, adversaries realizing chosen ciphertext attacks might not only know the access rights of the users (that is, the key indices of their keys), but could also exploit the fact that the same secret key is used several times. The CK-model was previously used in [7].

In this section, we prove that under CCA attacks the OK-model and the OU-model are weaker than the CK-model (cf. Fig. 2). We notice that using the **CKGen** oracle and the **Open** oracle we can simulate the behavior of every adversary in the other two models. Hence, CK-security implies OK-security and OU-security. Furthermore, under CPA the CK-model and the OU-model are equivalent due to the absence of the decryption oracle. The weakness of the OK-model under CPA can be argued from the corresponding result under CCA. All mentioned results hold for PE as well as for P-KEM. We show the separation



A barred arrow is a separation. A dashed arrow denotes obvious implication.

Fig. 2. Relation between different security models for PE and P-KEM under CCA attacks on the left and under CPA attacks on the right.

results for P-KEM, since the constructions in the proofs are a bit more involved in this case.

In order to consider the relations between the introduced security models $\mathrm{MOD} \in \{\mathrm{OK}, \mathrm{OU}, \mathrm{CK}\}$, we define sets of adversaries $\mathbf{A}^{\mathrm{MOD}}_{\text{P-KEM}}$. We replace the generic oracles available for $\mathcal{A} \in \mathbf{A}^{\mathrm{IND}}_{\text{P-KEM}}$ by oracles defined in Table 1. The corresponding security experiments are denoted by $\text{P-KEM}^{\mathrm{ATK,MOD}}_{\Pi,\mathcal{A}}(\lambda, \mathrm{des})$.

Definition 4.1. *Let* $\mathrm{MOD} \in \{\mathrm{OK}, \mathrm{OU}, \mathrm{CK}\}$, $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA}\}$, *and* $\mathcal{R}_{\Omega,\Sigma}$ *be a predicate family. A* P-KEM Π *with public index for* $\mathcal{R}_{\Omega,\Sigma}$ *is called secure in model* MOD *under attack* ATK *(also* MOD-ATK*-secure) if for every* $\mathrm{des} \in \Omega$ *and every ppt adversary* $\mathcal{A} \in \mathbf{A}^{\mathrm{MOD}}_{\text{P-KEM}}$*, the advantage* $\text{Adv-P-KEM}^{\mathrm{ATK,MOD}}_{\Pi,\mathcal{A}}(\lambda, \mathrm{des})$ *of* $\mathcal{A}$ *defined by* $2 \cdot \Pr\left[\text{P-KEM}^{\mathrm{ATK,MOD}}_{\Pi,\mathcal{A}}(\lambda, \mathrm{des}) = 1\right] - 1$ *is negligible.*

In the following subsections we also write MOD-secure instead of MOD-ATK-secure, when the attack scenario is obvious from the context.

4.1 OK-Security Does Not Imply OU-Security and CK-Security

In this subsection we construct an OK-secure scheme which is neither OU-secure nor CK-secure. We start from an OK-secure scheme and assume existence of pseudorandom functions (PRFs) [14].

Theorem 4.1. *Let* $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA}\}$, *and* $\mathcal{PRF}$ *be a family of* PRF*s. Suppose* Π *is an* OK-ATK*-secure* P-KEM *for predicate family* $\mathcal{R}_{\Omega,\Sigma}$*. Then, there exists an* OK-ATK*-secure* P-KEM *scheme* Π' *for* $\mathcal{R}_{\Omega,\Sigma}$ *which is neither* OU-ATK*-secure nor* CK-ATK*-secure.*

Proof. Let $\langle\cdot\rangle$ be any canonical representation of the master secret keys of scheme Π. W.l.o.g. we assume that for every $\mathrm{des} \in \Omega$ and every $(\mathrm{msk}, \mathrm{pp}_\kappa) \in \left[\mathrm{Setup}\left(1^\lambda, \mathrm{des}\right)\right]$ it holds $|\langle \mathrm{msk} \rangle| = \mathrm{MKLen}(\lambda)$ for some polynomial $\mathrm{MKLen}(\cdot)$. Π' is the same as Π with two extensions. The master secret key is extended by $\mathrm{PRF}\ f : \mathbb{X}_\kappa \mapsto \{0,1\}^{\mathrm{MKLen}(\lambda)}$. Every secret key is extended by the value *rand* which takes the value of $f(\mathrm{kInd})$ or the value of $f(\mathrm{kInd}) \oplus \langle \mathrm{msk} \rangle$ each with probability $1/2$.

Scheme Π' is trivially broken in the OU-model and in the CK-model, where the adversary can get several keys for the same key index and hence, learns msk. At the same time Π' remains OK-secure, since the adversary receives for every kInd either $f(\text{kInd})$ or $f(\text{kInd}) \oplus \langle \text{msk} \rangle$ and hence, due to the property of the PRF the values *rand* are useless for $\mathcal{A}$. □

The construction in the proof reveals the weakness of the OK-model even though the scheme is artificial. Namely, the OK-model does not cover the fact that several keys for the same key index exist. At least potentially, user secret keys for the same key index can leak more information about the master secret key than user secret keys for different key indices. Hence, already under CPA the OK-model makes unwarranted restrictions of adversarial abilities which might cause security issues. In [11,13], the authors present CPA-secure IBE schemes and explicitly prevent generation of different user secret keys for the same identity. For these purposes the master secret key is extended by a PRF F and the user secret key for kInd is generated using random bits $F(\text{kInd})$. The basic schemes without this modification are still secure in the OK-model, but are insecure in the OU-model. It is important to observe that this generic technique to achieve OU-security is insufficient if schemes with additional functionality such as key delegation are considered. Namely, delegation of the user secret keys requires generation of new (more restrictive) keys by the users.

4.2 OU-Security Does Not Imply OK-Security and CK-Security Under Chosen-Ciphertext Attack

In this subsection we consider only chosen-ciphertext attacks and construct an OU-secure scheme which is neither OK-secure nor CK-secure.

Theorem 4.2. *Suppose* Π *is an* OU-CCA*-secure* P-KEM *for predicate family* $\mathcal{R}_{\Omega,\Sigma}$ *and family of key spaces* $\mathcal{K} = \{\mathbb{K}_\lambda\}$*. Then, there exists an* OU-CCA*-secure* P-KEM *scheme* Π' *for* $\mathcal{R}_{\Omega,\Sigma}$ *and* $\mathcal{K}$ *which is neither* OK-CCA*-secure nor* CK-CCA*-secure.*

Proof. Let $\mathbb{K}_\lambda = \{0,1\}^{\text{KLen}(\lambda)}$ for polynomial $\text{KLen}(\lambda)$. In the construction of Π' we assume w.l.o.g. that for all λ, all $(\text{msk}, \text{pp}_\kappa) \in [\text{Setup}(1^\lambda, \text{des})]$, all $\text{kInd} \in \mathbb{X}_\kappa$, and all $\text{sk} \in [\text{KeyGen}(\text{msk}, \text{kInd})]$ it holds $|\langle \text{sk} \rangle| = \text{KLen}(\lambda)$, where $\langle \cdot \rangle$ is any canonical representation of the user secret keys. P-KEM scheme Π' is as follows:

- $\text{Setup}'(1^\lambda, \text{des}) = \text{Setup}(1^\lambda, \text{des})$.
- $\text{KeyGen}'(\text{msk}, \text{kInd})$: generate $\text{sk} \leftarrow \text{KeyGen}(\text{msk}, \text{kInd})$, choose a bit string $r \leftarrow \{0,1\}^{|\langle \text{sk} \rangle|}$, output $\text{sk}' := (\text{sk}, r)$.
- $\text{Encaps}'(\text{cInd})$: generate $(\text{CT}, \text{K}) \leftarrow \text{Encaps}(\text{cInd})$ set $\text{CT}' = 00 \| \text{CT}$ and output (CT', K).
- $\text{Decaps}'(\text{sk}', \text{CT}')$: parse $\text{sk}' = (\text{sk}, r)$ and $\text{CT}' = b_1 b_2 \| \text{CT}$ where $b_1, b_2 \in \{0,1\}$. Output $\text{Decaps}(\text{sk}, \text{CT})$ if $b_1 = b_2 = 0$; output r if $b_1 = 1 \wedge b_2 = 0$; output $r \oplus \langle \text{sk} \rangle$ if $b_1 = b_2 = 1$; and output $\perp$ otherwise.

Π' is OU-secure since using the decapsulation oracle the OU-adversary will be able to learn either r or $r \oplus \langle \mathrm{sk} \rangle$ which on their own are useless. This is due to the fact that every key is used only once in OU-model. In the other two security models the adversary can get every user secret key using only two decapsulation queries and can trivially break the scheme. □

Even though the scheme Π' in the proof is artificial, it reveals the main weakness of the OU-model under CCA. Namely, this model does not ensure that the decapsulation oracle does not leak partial information about the user secret key if queried with an ill-formed encapsulation. One general technique to achieve CK-security from OU-secure schemes is to rerandomize the user secret key for every decapsulation. However, schemes considered in the next subsection achieve CK-security without this costly modification.

4.3 Discussion and Recommendation

As mentioned in the introduction to this section most known PE schemes are proved secure in the OK-security model or in the OU-security model. The relation between OK-security and OU-security under CPA is quite simple. Hence, in this section we examine the CK-CCA-security of several known schemes.

It is important to observe that the correctness property of PE ensures that decryption of a correctly generated ciphertext using a proper key always results in the correct message. In contrast, for an ill-formed ciphertext decryption using different keys for the same key index might result in different outputs, which matters in the CK-security model. Many CCA-secure schemes perform consistency checks for the ciphertexts during decryption, which allow us to argue about their CK-security. However, for most known PE schemes for involved predicates the correct form of the ciphertexts with respect to the encryption algorithm cannot be efficiently checked. Hence, consistency checks do not provide a generic method to achieve CK-security.

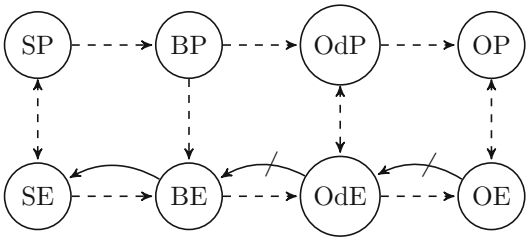
In the IBE scheme from [10] keys are unique by construction and hence, all three security notions are identical for this scheme. But usually IBE schemes do not have unique user secret keys (cf. [16,17,24]). In the basic scheme from [16] ill-formed ciphertexts are explicitly rejected by the decryption algorithm and the output is independent of the used key. The second scheme from [16] uses fresh randomness during the decryption and rejects ill-formed ciphertexts with overwhelming probability independently of the used key. The schemes in [17] reject ill-formed ciphertexts with overwhelming probability due to the authenticated symmetric encryption. Furthermore, the generic CPA-to-CCA transformations for attribute-based encryption (ABE) [25] and for predicate encryption [26] require the existence of verification algorithms which ensure that the output of the decryption algorithm is independent of the used secret key. Hence, all schemes considered so far are secure in the CK-security model. For the schemes from [11,20,25] it is not obvious how to argue from the original proofs whether the CK-security notion is satisfied or not. We leave this as an open question.

Due to the results of this section we encourage to use the CK-model under CCA in order to specify the security guarantees of the schemes precisely. If CPA attacks are considered we recommend to use the simpler OU-model.

5 When and How to Restrict Challenge Decryptions

Let us briefly recall four security notions for PKE identified and formalized in [5]. There are two dimensions in the definition of CCA-security regarding the restriction of adversaries to query the decryption of the challenge ciphertext. The first dimension specifies the style of restrictions, or rather how these queries are disallowed. The authors differentiate between the penalty style definition (denoted by P) and the exclusion style definition (denoted by E). So far we have used the former style in this work. That is, an adversary that violates restrictions of the security experiment is penalized at the end of the experiment (we model this using the BadQuery event). In the exclusion style definition, the set of adversaries is restricted from the beginning such that for every considered adversary the probability that the restrictions are violated is zero. Furthermore, one can disallow the adversary to query the decryption of the challenge ciphertext only in the second query phase (denoted by S) or in both query phases (denoted by B). This results in four security notions denoted by SP, SE, BP, and BE. For example, in the BP-model the output of the experiment is defined to be zero (adversary looses) if $\mathcal{A}$ queries the decryption of the challenge ciphertext CT^* in the first or in the second query phase. In contrast, in the BE-model we do not consider adversary $\mathcal{A}$, if the probability that $\mathcal{A}$ violates the restriction is non-zero. In order to prove that BE-security implies BP-security we have to show that given an adversary $\mathcal{A}$ with non-negligible advantage in the BP-model we can construct an adversary $\mathcal{A}'$ with non-negligible advantage in the BE-model. This is a non-trivial task, since in the first query phase the challenge ciphertext is not known. In contrast, SE-security trivially implies SP-security. Finally, security in a penalty style model implies security in the corresponding exclusion style model by design.

For PE, we can prove the same relations between the mentioned notions as for PKE which is not surprising and was stated without proof for IBE schemes in [5]. We refer to the full version [8] for details. For conventional KEMs two additional notions are considered in [5]. Namely, the first query phase can be completely dropped mainly due to the fact that the adversary cannot influence the generated challenge. This results in two additional security models denoted by OP and OE, where "O" stands for "One phase". For conventional KEM all six security notions are equivalent [5]. However, we show that P-KEM substantially differ from KEM, since the adversary might be able to influence the ciphertext index of the target encapsulation. We prove that, due to this property, the corresponding security notions are not all equivalent. First of all, because of the key generation oracle we consider two additional security notions, where the adversary has access to this oracle in both phases, but the decapsulation oracle is available only in the second query phase. The corresponding penalty style security model is denoted



The continuous arrows denote implications and the barred arrows denote separations. The dashed arrows denote trivial implications.

Fig. 3. Relation between different security notions for P-KEM.

by OdP and the exclusion style model is denoted by OdE. Here "Od" stands for "One decapsulation" phase.

We prove that the OdE and the OdP security notions are weaker than the BE security notion (cf. Fig. 3). The one-phase security notions OE and OP are even weaker. We also prove that the other four security notions (SP, SE, BP, and BE) are equivalent for P-KEM. Nevertheless, the reductions are not tight. The main difference between KEM and P-KEM is that whereas security of conventional KEM implies smoothness [5], this is not the case for P-KEM. Rather, BE-security implies that every ppt algorithm has only a negligible advantage in finding a ciphertext index with only few possible encapsulations. We call such a ciphertext index a *weak ciphertext* index.

The CCA-security experiment for all models is defined in Fig. 4. The oracles are as defined in Table 1 for CK-model, with restrictions of adversaries for every model as explained above (see the full version for the formal definition). Furthermore, we refer to the full version [8] for the separation result between OE and OdE.

OdE $\not\Rightarrow$ BE. Compared to BE-adversaries OdE-adversaries do not make decapsulation queries in the first query phase. Consider a predicate family $\mathcal{R}$ of prefix predicates with $\mathbb{X}_\kappa = \mathbb{Y}_\kappa = \{0,1\}^{\leq n}$, where $n = 2 \cdot \lambda$, that is $\mathrm{R}_\kappa(\mathrm{kInd}, \mathrm{cInd}) = 1$ if and only if kInd is a prefix of cInd. Note that CCA-secure P-KEM for $\mathcal{R}$ can be realized from hierarchical IBE [18]. We additionally require an injective one-way function family $\mathcal{F}$ (see e.g. [14] for formal definitions). Injectivity of $\mathcal{F}$ simplifies the proof, but is not necessary.

Theorem 5.1. *Suppose $\mathcal{F}$ is a family of injective one-way functions and Π is an* OdE*-secure* P-KEM *for predicate family $\mathcal{R}$ and a family of key spaces $\mathcal{K}$. Then, there exists a* P-KEM *Π' for $\mathcal{R}$ and $\mathcal{K}$ which is* OdE*-secure but not* BE*-secure. In particular, for every* $\mathrm{des} \in \Omega$ *and every ppt* $\mathcal{A} \in \mathbf{A}^{\mathrm{OdE}}_{\Pi',\text{P-KEM}}$ *there exist ppt adversaries* $\mathcal{B}, \mathcal{B}' \in \mathbf{A}^{\mathrm{OdE}}_{\Pi,\text{P-KEM}}$ *and a ppt inverter $\mathcal{I}$ for $\mathcal{F}$ such that*

$$\begin{aligned}\text{Adv-P-KEM}^{\mathrm{CCA,OdE}}_{\Pi',\mathcal{A}}(\lambda, \mathrm{des}) \leq\ & \text{Adv-P-KEM}^{\mathrm{CCA,OdE}}_{\Pi,\mathcal{B}}(\lambda, \mathrm{des}) + \text{Adv-Inv}_{\mathcal{F},\mathcal{I}}(\lambda) \\ & + 4 \cdot \lambda \cdot \text{Adv-P-KEM}^{\mathrm{CCA,OdE}}_{\Pi,\mathcal{B}'}(\lambda, \mathrm{des})\,.\end{aligned}$$

$\mathbf{P\text{-}KEM}^{\mathrm{CCA,MOD}}_{\Pi,\mathcal{A}}(\lambda, \mathrm{des})$:

$b \leftarrow \{0,1\}\,; S_1, S_2, S_k \leftarrow \emptyset; (\mathrm{msk}, \mathrm{pp}_\kappa) \leftarrow \mathrm{Setup}\left(1^\lambda, \mathrm{des}\right);$

$(\mathrm{cInd}^*, St) \leftarrow \mathcal{A}_1^{\mathbf{CKGen}(\cdot),\mathbf{Open}(\cdot),\mathbf{Decaps}_1(\cdot,\cdot)}\left(1^\lambda, \mathrm{pp}_\kappa\right);$

Output b if the output of $\mathcal{A}_1$ is $\perp$;

$(\mathrm{K}_0, \mathrm{CT}^*) \leftarrow \mathrm{Enc}\left(\mathrm{pp}_\kappa, \mathrm{cInd}^*\right); \mathrm{K}_1 \leftarrow \mathbb{K}_\lambda; \mathrm{K}^* := \mathrm{K}_b;$

$b' \leftarrow \mathcal{A}_2^{\mathbf{CKGen}(\cdot),\mathbf{Open}(\cdot),\mathbf{Decaps}_2(\cdot,\cdot)}\left(\mathrm{K}^*, \mathrm{CT}^*, St\right);$

Output : $\mathrm{MOD} \in \{\mathrm{SE}, \mathrm{BE}, \mathrm{OdE}, \mathrm{OE}\}$: return $b' = b$;

$\mathrm{MOD} \in \{\mathrm{SP}, \mathrm{OdP}, \mathrm{OP}\}$: return $b' = b \wedge (\mathrm{CT}^* \notin S_2)$;

$\mathrm{MOD} = \mathrm{BP}$: return $b' = b \wedge (\mathrm{CT}^* \notin S_1 \cup S_2)$.

S_i contains all encapsulations submitted to $\mathbf{Decaps}_i$.

Fig. 4. CCA-security experiment for different security notions of P-KEM.

Proof. Assume w.l.o.g. that $\mathcal{K} = \{\mathbb{K}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathbb{K}_\lambda = \{0,1\}^\lambda$ and that the encapsulations under cInd are of the form (cInd, ct). Π' constructed from Π as follows:

$\mathbf{Setup}'\left(1^\lambda, \mathrm{des}\right)$: Choose $(\mathrm{pp}_\kappa, \mathrm{msk}) \leftarrow \mathrm{Setup}\left(1^\lambda, \mathrm{des}\right)$, $r = (r_1, \ldots, r_\lambda) \leftarrow \{0,1\}^\lambda$. Set $\mathrm{cInd}_w := 1^\lambda \| r$ and for all $i \in [\lambda]$ denote $\mathrm{kInd}_i = \mathrm{cInd}_i = 1^i$. For every $i \in [\lambda]$ generate $(\mathrm{CT}_i, \mathrm{K}_i) \leftarrow \mathrm{Encaps}\left(1^\lambda, \mathrm{cInd}_i\right)$ until the i'th bit of K_i is equal r_i. Choose $\mathrm{K}_w \leftarrow \mathbb{K}_\lambda$, $f \leftarrow \mathcal{F}$, and compute $Y := f(r)$. Output msk and $\mathrm{pp}'_\kappa := (\mathrm{pp}_\kappa, f, \mathrm{K}_w, Y, \mathrm{CT}_1, \ldots, \mathrm{CT}_\lambda)$.

$\mathbf{KeyGen}'\left(\mathrm{pp}'_\kappa, \mathrm{msk}, \mathrm{kInd}\right)$: Output $\mathrm{sk} \leftarrow \mathrm{KeyGen}\left(\mathrm{pp}_\kappa, \mathrm{msk}, \mathrm{kInd}\right)$.

$\mathbf{Encaps}'\left(\mathrm{pp}'_\kappa, \mathrm{cInd}\right)$: If $\mathrm{cInd} = 1^\lambda \| r'$ and $f(r') = Y$ output K_w and $\mathrm{CT}' = 1 \| \left(\mathrm{cInd}, 1^\lambda\right)$. Otherwise compute $(\mathrm{K}, \mathrm{CT}) \leftarrow \mathrm{Encaps}\left(\mathrm{pp}_\kappa, \mathrm{cInd}\right)$ and output K and $\mathrm{CT}' := 0 \| \mathrm{CT}$.

$\mathbf{Decaps}'\left(\mathrm{pp}'_\kappa, \mathrm{CT}', \mathrm{sk}\right)$: Parse $\mathrm{CT}' = b \| (\mathrm{cInd}, ct)$. Output K_w if $b = 1$, $ct = 1^\lambda$, $\mathrm{cInd} = 1^\lambda \| r'$, and $f(r') = Y$. If $b = 1$ output $\perp$. Otherwise output $\mathrm{Decaps}\left(\mathrm{pp}_\kappa, (\mathrm{cInd}, ct), \mathrm{sk}\right)$.

Obviously, Π' is not BE-secure, since the weak ciphertext index $\mathrm{cInd}_w = 1^\lambda \| r$ can be revealed using the decapsulation oracle on CT_i's.

To show that Π' is OdE-secure, observe that, informally, the OdE-security of Π ensures that an OdE adversary against Π' cannot learn information from $\mathrm{CT}_i \in \mathrm{pp}'_\kappa$ in the first query phase without querying a key for a prefix of cInd_λ. However, if $\mathcal{A}$ queries such a key, she cannot use cInd_w for the challenge anymore. Next we sketch how to turn this intuition into a formal proof that Π' is OdE-secure. Given $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) \in \mathbf{A}^{\mathrm{OdE}}_{\Pi',\text{P-KEM}}$ we construct $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2) \in \mathbf{A}^{\mathrm{OdE}}_{\Pi,\text{P-KEM}}$ which extends the given public parameters, simulates $\mathcal{A}$ and exploits her success probability except for the case that $\mathcal{A}_1$ outputs $\mathrm{cInd}^* = \mathrm{cInd}_w$. Then, we prove that the probability for $\mathrm{cInd}^* = \mathrm{cInd}_w$ is negligible. Namely, we construct an inverter $\mathcal{I}$ for $\mathcal{F}$, which given the challenge (f, Y) generates the public parameters as defined in the scheme except for $\{\mathrm{CT}_i\}_{i\in[\lambda]}$ computed by $\mathrm{Encaps}\left(1^\lambda, \mathrm{cInd}_i\right)$.

$\mathcal{I}$ wins if $\mathcal{A}_1$ outputs the challenge index $\mathrm{cInd}^* = \mathrm{cInd}_w = 1^\lambda \| r'$, $f(r') = Y$. The last step is to prove that the probability for $\mathrm{cInd}^* = \mathrm{cInd}_w$ in the real experiment and in the experiment with public parameters as generated by $\mathcal{I}$ is the same except for negligible probability. For this step we consider hybrid distributions, one for each modification of a single CT_i which results in the reduction algorithm $\mathcal{B}'$. □

BE ⇒ SE. Finally we state our result that BE-security implies SE-security for P-KEMs. A proof for this result is given in the full version [8]

Theorem 5.2. *Suppose* Π *is a* BE*-secure* P-KEM *for predicate family* $\mathcal{R}_{\Omega,\Sigma}$. *Then,* Π *is* SE*-secure. In particular, for every* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) \in \mathbf{A}^{\mathrm{SE}}_{\text{P-KEM}}$ *and every* $\mathrm{des} \in \Omega$ *there exist* $\mathcal{A}', \mathcal{A}'' \in \mathbf{A}^{\mathrm{BE}}_{\text{P-KEM}}$ *such that*

$$\mathrm{Adv\text{-}P\text{-}KEM}^{\mathrm{CCA,SE}}_{\Pi,\mathcal{A}}(\lambda, \mathrm{des}) \leq (l+1) \cdot \mathrm{Adv\text{-}P\text{-}KEM}^{\mathrm{CCA,BE}}_{\Pi,\mathcal{A}'}(\lambda, \mathrm{des}) + l \cdot \sqrt{2 \cdot (l+1) \cdot \mathrm{Adv\text{-}P\text{-}KEM}^{\mathrm{CCA,BE}}_{\Pi,\mathcal{A}''}(\lambda, \mathrm{des})} \ ,$$

where $l = l(\lambda, \mathrm{des})$ *is the upper bound for the maximum number of decapsulation queries of* $\mathcal{A}_1$ *in* $\mathrm{P\text{-}KEM}^{\mathrm{CCA,SE}}_{\Pi,\mathcal{A}}(\lambda, \mathrm{des})$.

Even for smooth schemes the reduction given by this theorem is not tight. Namely, the security guarantees linearly decrease in the number of decapsulation queries of adversary in the first query phase even for smooth schemes. In contrast, the corresponding reduction for conventional KEMs from [5] is tight.

Summarizing our results, we see that the security notions SP, SE, BP, and BE are equivalent for P-KEMs. But some of the reductions are tight, e.g. Theorem 5.2. For the implication BP ⇒ SP a tight reduction can be presented for smooth schemes. To the best of our knowledge all practical predicate encryption schemes are smooth and hence, we could also use the BP-model for these schemes. Since the probability for the event $\mathrm{CT}^* \in S_1$ can always be estimated by $l \cdot \mathrm{Smth}_\Pi(\lambda, \mathrm{des})$, we do not really get any advantage from this model. Hence, we recommend to use the SE-model.

References

1. Attrapadung, N.: Dual system encryption via doubly selective security: framework, fully secure functional encryption for regular languages, and more. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 557–577. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_31
2. Attrapadung, N., et al.: Relations among notions of security for identity based encryption schemes. In: Correa, J.R., Hevia, A., Kiwi, M. (eds.) LATIN 2006. LNCS, vol. 3887, pp. 130–141. Springer, Heidelberg (2006). https://doi.org/10.1007/11682462_16
3. Attrapadung, N., Imai, H.: Dual-policy attribute based encryption: simultaneous access control with ciphertext and key policies. IEICE Trans. **93–A**(1), 116–125 (2010)

4. Barbosa, M., Farshim, P.: On the semantic security of functional encryption schemes. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 143–161. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_10
5. Bellare, M., Hofheinz, D., Kiltz, E.: Subtleties in the definition of IND-CCA: when and how should challenge decryption be disallowed? J. Cryptol. **28**(1), 29–48 (2015)
6. Bellare, M., O'Neill, A.: Semantically-secure functional encryption: possibility results, impossibility results and the quest for a general definition. In: Abdalla, M., Nita-Rotaru, C., Dahab, R. (eds.) CANS 2013. LNCS, vol. 8257, pp. 218–234. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-02937-5_12
7. Blömer, J., Liske, G.: Construction of fully CCA-secure predicate encryptions from pair encoding schemes. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 431–447. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_25
8. Blömer, J., Liske, G.: Subtleties in security definitions for predicate encryption with public index. IACR Cryptology ePrint Archive 2017, 453 (2017). http://eprint.iacr.org/2017/453
9. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. **36**(5), 1301–1328 (2007)
10. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. SIAM J. Comput. **32**(3), 586–615 (2003)
11. Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), pp. 647–657. IEEE Computer Society (2007)
12. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
13. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: 40th Annual ACM Symposium on Theory of Computing, pp. 197–206. ACM (2008)
14. Goldreich, O.: The Foundations of Cryptography - Basic Tools, vol. 1. Cambridge University Press, Cambridge (2004)
15. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. CRC Press, Boca Raton (2014)
16. Kiltz, E., Galindo, D.: Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. Theor. Comput. Sci. **410**(47–49), 5093–5111 (2009)
17. Kiltz, E., Vahlis, Y.: CCA2 secure IBE: standard model efficiency through authenticated symmetric encryption. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 221–238. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79263-5_14
18. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_27
19. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_12
20. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_11

21. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010). https://eprint.iacr.org/2010/556
22. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
23. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5
24. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_7
25. Yamada, S., Attrapadung, N., Hanaoka, G., Kunihiro, N.: Generic constructions for chosen-ciphertext secure attribute based encryption. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 71–89. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_5
26. Yamada, S., Attrapadung, N., Santoso, B., Schuldt, J.C.N., Hanaoka, G., Kunihiro, N.: Verifiable predicate encryption and applications to CCA security and anonymous predicate authentication. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 243–261. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_15

Code-Based Key Encapsulation from McEliece's Cryptosystem

Edoardo Persichetti(✉)

Florida Atlantic University, Boca Raton, USA
epersichetti@fau.edu

Abstract. In this paper we show that it is possible to extend the framework of Persichetti's Nierreiter-based KEM [11] and create a secure KEM based on the McEliece protocol. This provides greater flexibility in the application of coding theory as a basis for cryptographic purposes.

1 Introduction

A *Hybrid Encryption* scheme is a cryptographic protocol that uses public-key encryption as means to securely exchange a key, while delegating the task of encrypting the body of the message to a symmetric scheme. The public-key component is known as *Key Encapsulation Mechanism (KEM)*. The first code-based KEM, utilizing the Niederreiter framework [9], was presented by Persichetti in [11] and successively implemented in [3]. In this paper, we expand on Persichetti's work and prove that if we use the McEliece approach [7] we are still able to obtain a secure KEM. This is a novel construction, with a great potential impact, especially considering NIST's recent call for papers for secure post-quantum primitives [1].

2 Preliminaries

2.1 The McEliece Cryptosystem

We consider here a more "modern" version compared to McEliece's original cryptosystem [7]. In the description that we use (Table 2, Appendix A), we consider families of codes to which is possible to associate an efficient decoding algorithm; we denote this with Decode_{Δ}, where Δ is a description of the selected code that depends on the specific family considered. For instance, in the case of binary Goppa codes, the associated algorithm is Patterson's algorithm [10] and Δ is given by a Goppa polynomial $g(x)$ and its support $(\alpha_1, \ldots, \alpha_n)$. For MDPC codes [8], decoding is given by Gallager's bit-flipping algorithm [6] and Δ is a sparse parity-check matrix H for the code. Also, we denote with $\mathbb{W}_{q,n,w}$ the set of words of $\mathbb{F}_q^n$ with Hamming weight w.

The security of the scheme follows from the two following computational assumptions.

J. Blömer et al. (Eds.): MACIS 2017, LNCS 10693, pp. 454–459, 2017.
https://doi.org/10.1007/978-3-319-72453-9_36

Assumption 1 (Indistinguishability). *The $k \times n$ matrix G output by* KeyGen *is computationally indistinguishable from a same-size uniformly chosen matrix.*

Assumption 2 (Decoding Hardness). *Let G be a generator matrix for an $[n, k]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$ and y a word of $\mathbb{F}_q^n$. It is hard to find a codeword $c \in \mathcal{C}$ with $d(c, y) \leq w$.*

Assumption 2 is also known as the General Decoding Problem (GDP), which was proved to be NP-complete in [2], and it is believed to be hard on average, and not just on the worst-case instances (see for example Sendrier [12]).

2.2 Encapsulation Mechanisms and the Hybrid Framework

A Key Encapsulation Mechanism (KEM) is essentially a Public-Key Encryption scheme (PKE), with the exception that the encryption algorithm takes no input apart from the public key, and returns a pair (K, ψ_0). The string K has fixed length ℓ_K, specified by the KEM, and ψ_0 is an "encryption" of K in the sense that $\mathsf{Dec}_{\mathsf{sk}}(\psi_0) = K$. The key K produced by the KEM is then passed on to a Data Encapsulation Mechanism (DEM), which is in charge of encrypting the actual message. The formulation of a DEM, that normally comprises additional tools for security such as Message Authentication Codes (MAC), is outside the scope of this paper, and we refer the reader to [5] for more details.

A KEM is required to be *sound* for at least all but a negligible portion of public key/private key pairs, that is, if $\mathsf{Enc}_{\mathsf{pk}}(\ ) = (K, \psi_0)$ then $\mathsf{Dec}_{\mathsf{sk}}(\psi_0) = K$ with overwhelming probability.

The security notions for a KEM are similar to the corresponding ones for PKE schemes. The one we are mainly interested in (representing the highest level of security) is IND-CCA, which we describe below.

Definition 1. *The adaptive Chosen-Ciphertext Attack game for a KEM proceeds as follows:*

1. *Query a key generation oracle to obtain a public key* pk.
2. *Make a sequence of calls to a decryption oracle, submitting any string ψ_0 of the proper length. The oracle will respond with $\mathsf{Dec}_{sk}^{KEM}(\psi_0)$.*
3. *Query an encryption oracle. The oracle runs Enc_{pk}^{KEM} to generate a pair $(\tilde{K}, \tilde{\psi_0})$, then chooses a random $b \in \{0, 1\}$ and replies with the "challenge" ciphertext $(K^*, \tilde{\psi_0})$ where $K^* = \tilde{K}$ if $b = 1$ or K^* is a random string of length ℓ_K otherwise.*
4. *Keep performing decryption queries. If the submitted ciphertext is ψ_0^*, the oracle will return $\perp$.*
5. *Output $b^* \in \{0, 1\}$.*

The adversary succeeds if $b^* = b$. More precisely, we define the advantage *of $\mathcal{A}$ against KEM as*

$$\mathsf{Adv}_{KEM}(\mathcal{A}, \lambda) = \left| Pr[b^* = b] - \frac{1}{2} \right|. \tag{1}$$

We say that a KEM is secure if the advantage Adv_{KEM} of any polynomial-time adversary $\mathcal{A}$ in the above CCA attack model is negligible.

It has then been proved that, given a CCA adversary $\mathcal{A}$ for the hybrid scheme (HY), there exist an adversary $\mathcal{A}_1$ for KEM and an adversary $\mathcal{A}_2$ for DEM running in roughly the same time as $\mathcal{A}$, such that for any choice of the security parameter λ we have $\mathsf{Adv}_{\mathsf{HY}}(\mathcal{A}, \lambda) \leq \mathsf{Adv}'_{\mathsf{KEM}}(\mathcal{A}_1, \lambda) + \mathsf{Adv}_{\mathsf{DEM}}(\mathcal{A}_2, \lambda)$. See Cramer and Shoup [5, Theorem 5] for a complete proof.

3 The New KEM Construction

The KEM we present here follows closely the McEliece framework, and is thus based on the hardness of GDP. Note that, compared to the original PKE, a slight modification is introduced in the decryption process. As we will see later, this is necessary for the proof of security. The ephemeral key K is obtained via a Key Derivation Function KDF (see Appendix B).

Table 1. The McEliece KEM.

Setup	Fix public system parameters $q, n, k, w \in \mathbb{N}$, then choose a family $\mathcal{F}$ of w-error-correcting $[n, k]$ linear codes over $\mathbb{F}_q$
KeyGen	Choose a code $\mathcal{C} \in \mathcal{F}$ with code description Δ and compute a generator matrix G. Generate a random $s \xleftarrow{\$} \mathbb{F}_q^k$. Public key is G and private key is (Δ, s)
Enc	On input a public key G choose random words $x \in \mathbb{F}_q^k$ and $e \in \mathbb{W}_{q,n,w}$, then compute $K = \mathsf{KDF}(x\|\|e, \ell_K)$, $\psi_0 = xG + e$ and return the key/ciphertext pair (K, ψ_0)
Dec	On input a private key Δ and a ciphertext ψ_0, compute $\mathsf{Decode}_\Delta(\psi_0)$. If the decoding succeeds, use its output (x, e) to compute $K = \mathsf{KDF}(x\|\|e, \ell_K)$. Otherwise, set $K = \mathsf{KDF}(s\|\|\psi_0, \ell_K)$. Return K

If the ciphertext is correctly formed, decoding will always succeed, hence the KEM is perfectly sound. Furthermore, it is possible to show that, even if with this formulation $\mathsf{Dec}^{\mathsf{KEM}}$ never fails, there is no integrity loss in the hybrid encryption scheme thanks to the check given by the MAC (Table 1).

We prove the security of the KEM in the following theorem.

Theorem 1. *Let $\mathcal{A}$ be an adversary in the random oracle model for the Niederreiter KEM as in Definition 1. Let θ be the running time of $\mathcal{A}$, n_{KDF} and n_{Dec} be two bounds on, respectively, the total number of random oracle queries and the total number of decryption queries performed by $\mathcal{A}$, and set $N = q^k \cdot |\mathbb{W}_{q,n,w}|$. Then there exists an adversary $\mathcal{A}'$ for GDP such that $Adv_{KEM}(\mathcal{A}, \lambda) \leq Adv_{GDP}(\mathcal{A}', \lambda) + n_{Dec}/N$. The running time of $\mathcal{A}'$ will be approximately equal to θ plus the cost of n_{KDF} matrix-vector multiplications and some table lookups.*

Proof. We replace KDF with a random oracle $\mathcal{H}$ mapping elements of the form $(x, e) \in \mathbb{F}_q^k \times \mathbb{W}_{q,n,w}$ to bit strings of length ℓ_K. To prove our claim, we proceed as follows. Let's call G_0 the original attack game played by $\mathcal{A}$, and S_0 the event that $\mathcal{A}$ succeeds in game G_0. We define a new game G_1 which is identical to G_0 except that the game is halted if the challenge ciphertext $\psi_0^* = x^*G + e^*$ obtained when querying the encryption oracle had been previously submitted to the decryption oracle: we call this event F_1. Since the number of valid ciphertexts is N, we have $\mathsf{Pr}[\mathsf{F}_1] \leq n_{\mathsf{Dec}}/N$. It follows that $\left|\mathsf{Pr}[\mathsf{S}_0] - \mathsf{Pr}[\mathsf{S}_1]\right| \leq n_{\mathsf{Dec}}/N$, where S_1 is the event that $\mathcal{A}$ succeeds in game G_1. Next, we define game G_2 which is identical to G_1 except that we generate the challenge ciphertext ψ_0^* at the beginning of the game, and we halt if $\mathcal{A}$ ever queries $\mathcal{H}$ at $(x^*||e^*)$: we call this event F_2. By construction, since $\mathcal{H}(x^*||e^*)$ is undefined, it is not possible to tell whether $K^* = K$, thus we have $\mathsf{Pr}[\mathsf{S}_2] = 1/2$, where S_2 is the event that $\mathcal{A}$ succeeds in game G_2. We obtain that $\left|\mathsf{Pr}[\mathsf{S}_1] - \mathsf{Pr}[\mathsf{S}_2]\right| \leq \mathsf{Pr}[\mathsf{F}_2]$ and we just need to bound $\mathsf{Pr}[\mathsf{F}_2]$.

We now construct an adversary $\mathcal{A}'$ against GDP. $\mathcal{A}'$ interacts with $\mathcal{A}$ and is able to simulate the random oracle and the decryption oracle with the help of two tables T_1 and T_2, initially empty, as described below.

Key Generation: On input the instance (G, y^*, w) of GDP, return the public key $\mathsf{pk} = G$.

Challenge Queries: When $\mathcal{A}$ asks for the challenge ciphertext:

1. Generate a random string K^* of length ℓ_K.
2. Set $\psi_0^* = y^*$.
3. Return the pair (K^*, ψ_0^*).

Random Oracle Queries: Upon $\mathcal{A}$'s random oracle query $(x, e) \in \mathbb{F}_q^k \times \mathbb{W}_{q,n,w}$:

1. Look up (x, e) in T_1. If (x, e, y, K) is in T_1 for some y and K, return K.
2. Compute $y = xG + e$.
3. If $y = y^*$ then $\mathcal{A}'$ outputs $c = xG$ and the game ends.
4. Look up y in T_2. If (y, K) is in T_2 for some K (i.e. the decryption oracle has been evaluated at y), return K.
5. Set K to be a random string of length ℓ_K and place (x, e, y, K) in table T_1.
6. Return K.

Decryption Queries: Upon $\mathcal{A}$'s decryption query $y \in \mathbb{F}_q^n$:

1. Look up y in T_2. If (y, K) is in T_2 for some K, return K.
2. Look up y in T_1. If (x, e, y, K) is in T_1 for some x, e and K (i.e. the random oracle has been evaluated at (x, e) such that $y = xG + e$), return K.
3. Generate a random string K of length ℓ_K and place the pair (y, K) in T_2.
4. Return K.

Note that, in both random oracle and decryption queries, we added the initial steps to guarantee the integrity of the simulation, that is, if the same value is queried more than once, the same output is returned. A fundamental issue is that it is impossible for the simulator to determine if a word is decodable or not. If the decryption algorithm returned $\perp$ if and only if a word was not decodable, then it would be impossible to simulate decryption properly. We have resolved this problem by insisting that the KEM decryption algorithm always outputs a hash value. With this formulation, the simulation is flawless and $\mathcal{A}'$ outputs a solution to the GDP instance with probability equal to $\mathsf{Pr}[\mathsf{F}_2]$. □

4 Conclusions

In this paper, we have introduced a key encapsulation method based on the McEliece cryptosystem. This novel approach enjoys a simple construction and a tight security proof as for the case of the Niederreiter KEM presented in [11]. We believe that our new construction will offer an important alternative while designing quantum-secure cryptographic primitives.

A The McEliece Cryptosystem

Table 2. The McEliece cryptosystem.

Setup	Fix public system parameters $q, n, k, w \in \mathbb{N}$, then choose a family $\mathcal{F}$ of w-error-correcting $[n, k]$ linear codes over $\mathbb{F}_q$
K	$\mathsf{K}_{\mathsf{publ}}$ the set of $k \times n$ matrices over $\mathbb{F}_q$
	$\mathsf{K}_{\mathsf{priv}}$ the set of code descriptions for $\mathcal{F}$
P	The vector space $\mathbb{F}_q^k$
C	The vector space $\mathbb{F}_q^n$
KeyGen	Generate at random a code $\mathcal{C} \in \mathcal{F}$ given by its code description Δ and compute a public[a] generator matrix G. Publish the public key $G \in \mathsf{K}_{\mathsf{publ}}$ and store the private key $\Delta \in \mathsf{K}_{\mathsf{priv}}$
Enc	On input a public key $G \in \mathsf{K}_{\mathsf{publ}}$ and a plaintext $\phi = x \in \mathsf{P}$, choose a random error vector $e \in \mathbb{W}_{q,n,w}$, then compute $y = xG + e$ and return the ciphertext $\psi = y \in \mathsf{C}$
Dec	On input the private key $\Delta \in \mathsf{K}_{\mathsf{priv}}$ and a ciphertext $\psi \in \mathsf{C}$, compute $\mathsf{Decode}_{\Delta}(\psi)$. If the decoding succeeds, return its output $\phi = x$. Otherwise, output $\perp$

[a] While the original version proposes to use scrambling matrices S and P (see [9] for details), this is not necessary and alternative methods can be used, depending on the chosen code family.

B Other Cryptographic Tools

In this section we introduce another cryptographic tool that we need for our construction.

Definition 2. *A* Key Derivation Function (KDF) *is a function that takes as input a string x of arbitrary length and an integer $\ell \geq 0$ and outputs a bit string of length ℓ.*

A KDF is modelled as a random oracle, and it satisfies the *entropy smoothing* property, that is, if x is chosen at random from a high entropy distribution, the output of KDF should be computationally indistinguishable from a random length-ℓ bit string.

Intuitively, a good choice for a KDF could be a hash function with a variable (arbitrary) length output, such as the new SHA-3, Keccak [4].

References

1. National Institute of Standards and Technology: Call for proposals, December 2016. http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf
2. Berlekamp, E., McEliece, R., van Tilborg, H.: On the inherent intractability of certain coding problems. IEEE Trans. Inf. Theory **24**(3), 384–386 (1978)
3. Bernstein, D.J., Chou, T., Schwabe, P.: McBits: fast constant-time code-based cryptography. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 250–272. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40349-1_15
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: http://keccak.noekeon.org/
5. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. **33**(1), 167–226 (2004)
6. Gallager, R.G.: Low-Density Parity-Check Codes. MIT Press, Cambridge (1963)
7. McEliece, R.: A public-key cryptosystem based on algebraic coding theory. Technical report, NASA (1978)
8. Misoczki, R., Tillich, J.-P., Sendrier, N., Barreto, P.S.L.M.: MDPC-McEliece: new McEliece variants from moderate density parity-check codes. Cryptology ePrint Archive, Report 2012/409 (2012)
9. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Probl. Control Inf. Theory **15**(2), 159–166 (1986)
10. Patterson, N.: The algebraic decoding of Goppa codes. IEEE Trans. Inf. Theory **21**(2), 203–207 (1975)
11. Persichetti, E.: Secure and anonymous hybrid encryption from coding theory. In: Gaborit, P. (ed.) PQCrypto 2013. LNCS, vol. 7932, pp. 174–187. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38616-9_12
12. Sendrier, N.: The tightness of security reductions in code-based cryptography. In: IEEE Information Theory Workshop (ITW), pp. 415–419, October 2011

Author Index