Transgressive Computing 2006

A conference in honor of Jean Della Dora

24-26 April 2006, Facultad de Ciencias, Universidad de Granada, España

Proceedings of Transgressive Computing 2006: a conference in honor of Jean Della Dora.
Editor: Jean-Guillaume Dumas, Université J. Fourier, Grenoble, France.

ISBN: X-XXXXX-XXX-X

Preface

From Wikipedia, the free encyclopedia, transgression may be:

- A legal transgression, a crime.
- A social transgression, violating a norm.
- Transgression (LDS theology), a violation of religious law without the perpetrator's understanding.
- Transgression (geology), an event during which sea level rises relative to the land, resulting in coastal flooding.
- Transgression (album), a 2005 album by the Los Angeles metal group Fear Factory.
- Transgression (novel), a science fiction novel by Randall Ingermanson.
- Transgressive art, a type of art that goes against basic norms or mores.
- Transgressional fiction, a form of literature.

By analogy, we can deduce that Transgressive computing is a type of computing that goes against basic norms or mores. To my knowledge (Google based), the use of transgressive computing is almost completely new (only one other reference).

Hence this conference is surely the first on this topics (and certainly not the last). Why such a title for a conference in honor of professor Jean Della Dora? What are these basic norms that are being transgressed? What kinds of trangressions? Is Jean Della Dora really a transgressive scientist: a type of scientist who goes against basic norms or mores?

Once again, from Wikipedia, the free encyclopedia, the term mores as used in sociology is a plural noun. The Latin singular, which is not used in English, is mos. The English word morality comes from the same root, as does the noun moral, which can mean the 'core meaning of a story'. Mores are strongly held norms or customs. These derive from the established practices of a society rather than its written laws. Taboos form the subset of mores that forbid a society's most outrageous behaviours, such as incest and murder in many societies. Usually these are formalized in some kind of moral code, e.g. commandments. Most sociologists reject the thesis that the formalization matters as much as the informal social response of disgust and isolation

of offenders. An example of a more might be someone picking their nose; which, although harmless, is widely considered as disgusting to the general populace and goes against the normal. However, constant exposure to social mores is thought by some to lead to development of an individual moral core, which is pre-rational and consists of a set of inhibitions that cannot be easily characterized except as potential inhibitions against taking opportunities that the family or society does not consider desirable. These in turn cannot be easily separated from individual opinions or fears of getting caught.

Looking for established practices, taboos or commandments in science, even in computing science is more that can be done in this short text. However, by just asking the question, we admit that such behaviours may exist in our field. Let me take as example the absolute necessity of a mathematical model. We, as computing scientists, do believe that we understand a natural phenomenon once we have a set of equations that "represents" it. And generally we are no longer interested in the natural phenomenon: we just work with the equations. They ARE the object under study. This is more than an established practice, more than a commandments, indeed we cannot leave without a mathematical model. Could we imagine a world without mathematical models. Are there other ways of modeling than with mathematics.

Moreover do we have finished our work as soon as the equations have been found. During long time, studying numerically the equations and discussing the quality of the results and the adequation with the natural phenomenon was like picking his nose in public for mathematicians.

For long our models have exclusively been built with partial differential equations. Discrete mathematics were so simple that established mathematicians did not consider them as real science: discrete mathematics based models would not be smart enough to model complicated phenomena. Working directly with discrete models rather than with continuous models was such a strong taboo that, in this new century, they are still considered as second class citizens.

Jean Della Dora is that kind of scientists who do not hesitate to pick their nose if they believe that science will progress from this. From that perspective, he certainly deserves such a conference.

But Jean Della Dora is not only a transgressive scientist, he has been also a transversal mathematician: numerical analysis (computing eigenvalues, Padé-Hermite approximation, numerical integration of ordinary differential equations), computer algebra

(differential equations, difference equations, algebraic equations and control), parallel algorithms and more recently hybrid systems with application in molecular biology and genetic networks.

In all these various domains, his unquenchable curiosity and original glance produced new trends of research: a border has been transgressed and the road was free. For years his colleagues and doctoral students were illuminated by this unique desire to understand better and more, to go further, to traverse new regions and transgress intellectual borders.

Such a behaviour has also influenced deeply his teaching and administrative activities. Jean Della Dora was the founder of many new courses and research teams. He has always been involved in the development of the Grenoble community in scientific computing, proceeding the path of his master Professor Noël Gastinel.

In the following texts, almost all the domains in which Jean Della Dora has been active are addressed. Even if he is no longer working in the field, his ideas are still valid and alive. Such a book will be useful as a reference for the included original contributions but also as a landmark by the diversity of the themes together with a unique way of approaching research.

Professor Jean Della Dora deserves more than that. He deserves our gratitude and recognition.

Sophia Antipolis, March 25, 2006.
Michel Cosnard.
Professor at École Polytechnique de l'Université de Nice - Sophia Antipolis.
Head of Sophia Antipolis INRIA Resarch Unit.

Contents

Extended abstracts

Program

Monday 24th	
8h00–9h00	Registration
9h00–9h25	**Opening**
9h25–9h50	Cung p.131
9h50–10h15	Takesue p.365
10h15–10h40	Vincent p.387
10h40–11h00	**Coffee**
11h00–11h50	Invited talk **C. Brézinski**
11h50–12h15	Krupchyk p.411
12h15–12h40	Shemyakova p.435
12h40–13h05	Mazure p.311
13h05–16h00	**Lunch**
16h00–16h50	Invited talk **J-P. Ramis**
16h50–17h15	Fauvet p.231
17h15–17h40	Chen p.93
17h40–18h05	Foursov p.257
18h05–18h25	**Coffee**
18h25–18h50	Rosaev p.429
18h50–19h15	Benmakrouha p.63
19h15–19h40	Fortuny Ayuso p.247
19h40–20h05	Hespel p.271

Tuesday 25th	
8h00–9h00	Registration
9h00–9h50	Invited talk **L. Brénig**
9h50–10h15	Falcón Ganfornina p.213
10h15–10h40	Martín-Morales p.303
10h40–11h00	**Coffee**
11h00–11h50	Invited talk **T. Recio**
11h50–12h15	Corless p.405
12h15–12h40	Mezzarobba p.327
12h40–13h05	Moreno Maza p.419
13h05–16h00	**Lunch**
16h00–16h50	Talk by **Jean Della Dora**
16h50–17h15	Boulier p.79
17h15–17h40	Dahan p.149
17h40–18h05	Tabera p.357
18h05–18h25	**Coffee**
18h25–18h50	Zhou p.443
18h50–19h15	Dumas p.185
19h15–19h40	El Ghazi p.203
19h40–20h05	Bouhamidi p.71
21h30	**Banquet**

Wednesday 26th	
8h00–9h00	Registration
9h00–9h50	Invited talk **E. Asarin**
9h50–10h15	Tonnelier p.381
10h15–10h40	Hildebrand p.287
10h40–11h00	**Coffee**
11h00–11h50	Invited talk **A. Carbone**
11h50–12h15	Cheptea p.105
12h15–12h40	Benjumea p.53
12h40–13h05	Diaz-Toca p.169
13h05–15h00	**Lunch**
15h00–15h50	Invited talk **S. Watt**
15h50–16h15	Cojocaru p.115
16h15–16h40	Liang p.415
16h40–17h05	Smirnova p.339
17h05–17h30	Brezinski p.401
20h00	**Opera**

Thursday 27th	
14h00-17h00	Alhambra Gardens visit
17h00-20h00	Alhambra Nasrides Palaces visit

SUPPORTS

Facultad de Ciencias de la Universidad de Granada
http://www.ugr.es/~decacien

Universidad de Granada
http://www.ugr.es

Université Joseph Fourier, Grenoble
http://www.ujf-grenoble.fr

Institut National Polytechnique de Grenoble
http://www.inpg.fr

Institut d'Informatique et de Mathématiques Appliquées de Grenoble
http://www.imag.fr

Ontario Research Center for Computer Algebra
http://www.orcca.org

Institut National de Recherche en Informatique et en Automatique
http://www.inria.fr

Laboratoire de Modélisation et Calcul
http://www-lmc.imag.fr

ORGANIZATION

- Organizing committee

Jean-Guillaume Dumas	Grenoble, France	Editor
José Gómez-Torrecillas	Granada, España	Local Arrangements
Françoise Jung	Grenoble, France	Treasurer
François Lemaire	Lille, France	Web master
Francisco-Javier Lobillo-Borrero	Granada, España	Local Arrangements
Marc Moreno-Maza	London, Canada	General Chair
Tomás Recio-Muñiz	Santander, España	Program Chair
Évelyne Tournier	Grenoble, France	Publicity

- Program committee

Bernhard Beckermann	Lille, France
Léon Brenig	Bruxelles, Belgique
Claude Brezinski	Lille, France
Bernard Brogliato	Montbonnot, France
José M. Cano Torres	Valladolid, España
Francisco-Jesús Castro-Jiménez	Sevilla, España
Guoting Chen	Lille, France
Robert M. Corless	London, Canada
Jean-Guillaume Dumas	Grenoble, France
Dominique Duval	Grenoble, France
Roderick Edwards	Victoria, Canada
Mark Giesbrecht	Waterloo, Canada
José Gómez-Torrecillas	Granada, España
Aziz Hilali	Rabat, Maroc
Françoise Jung	Grenoble, France
Lila Kari	London, Canada
François Lemaire	Lille, France
Francisco-Javier Lobillo-Borrero	Granada, España
Marc Moreno-Maza	London, Canada
Luis Miguel Pardo	Santander, España
Michel Petitot	Lille, France
Tomás Recio-Muñiz	Santander, España
Jean-Louis Roch	Grenoble, France
B. David Saunders	Delaware, USA
Éric Schost	Polytechnique, France
Évelyne Tournier	Grenoble, France
Gilles Villard	Lyon, France
Stephen M. Watt	London, Canada

Chapter 1

Invited talks

Dynamical systems
An algorithmic point of view

Jean Della Dora Aude Maignan Laurent Tournier

Abstract

The motivation to study dynamical systems is of course to get the more information we can about their phase portraits. To face this general problem, an interesting way seems promising, what we call here the concept of algorithmic system theory. At the basis of this concept is the analysis of several classes of systems, coming from different area (notably biology), which could be considered as elementary units of a bigger theory. What we present in this extended abstract is a couple of reflexions inspired by the work of our team over the past years.

1 Introduction

From an applied point of view, dynamical system theory presently oscillates between a deep theory with a sound mathematical foundation, but restricted to low dimensional systems, and a growing bulk of knowledge associated to higher dimensional systems using a numerical point of view, based on simulation and model reduction. In any case the algorithmic point of view is poorly taken into acount.

For several years we have tried to understand what could be such a point of view, and what computer algebra can bring to the subject. The result of our investigation is described into four thesis [1, 2, 3, 4] and several papers. In this paper we try to synthetise what we have learned and what could be an Algorithmic System Theory (AST).
Let's consider a dynamical system defined by a system of autonomous ODE:

$$X' = F(X)$$

where $X \in \Omega$, $\Omega \subset \mathbb{R}^n$. What is important is the knowledge of the phase portrait. In a certain sense it's the description of the behaviour of **all** the solutions of the dynamical system. This is a clear distinction with numerical integration of ODE whose primary goal is the construction of a particular solution of an initial value problem (IVP). Using simulation to build a phase portrait is so equivalent to the integration of a great number of IVP over a long period of time. But in any case, and this is a key point: simulation is not dedicated to *prove* something about an ODE.

On the other side, it's clear that the mathematical results that have been accumulated on low dimensional systems are difficult to apply on higher dimensional ones. The problem

is that huge dimensional dynamical systems are numerous in applied mathematics. Let's give an example :

Example 1.1. The numerical integration of partial differential equations (PDE) is a central subject of numerical analysis. In particular, the numerical integration of evolution equations is tremendously important. But behind the construction of a PDE, there exists a hypothesis named the continuity hypothesis. In an elementary volume, some equations for a bilan are established. Then it is argued that if the volumes decrease to zero, some limit exists and the PDE is established. This is very clear from a mathematical point of view but this is a perfect example of questionable physical point of view.
What is also very strange is that for numerical integration it's absolutly necessary to "come back" to a localized point of view and put the problem into an ODE framework ...
To be more precise let's take the example of the one-dimensional diffusion of heat:

$$\begin{cases} \dfrac{\partial u}{\partial t} - a\Delta u = f \\ u(0,x) = u^0(x) \end{cases} , \quad x \in [0,1]$$

with Dirichlet boundary conditions: $u(t,0) = \theta_0(t)$ and $u(t,1) = \theta_1(t)$.
a is a positive real, f, θ_0, θ_1 and u^0 are given functions and $u = u(x,t)$ is the possible solution of the PDE problem.
In order to study this problem we will use a discretization of the PDE. Let Δx be a positive real constant such that $\frac{1}{\Delta x}$ is an integer noted N. For $i \in \{0,\dots,N\}$, we consider $x_i = i\Delta x$. The x_i are a subdivision of the segment $[0,1]$. Then we pose the functions:

$$\begin{cases} u_0(t) = u(x_0,t) = u(0,t) = \theta_0(t) \\ u_i(t) = u(x_i,t) \,, \quad \text{for } i \in \{1,\dots,N-1\} \\ u_N(t) = u(x_N,t) = u(1,t) = \theta_1(t) \end{cases}$$

Then we discretize the laplacian term using a centered diagram:

$$\forall\, 1 \le i \le N-1\,, \quad \frac{\partial^2 u}{\partial x^2}(x_i,t) \simeq \frac{u(x_{i-1},t) - 2u(x_i,t) + u(x_{i+1},t)}{\Delta x^2}$$

We obtain the ordinary differential equations:

$$\forall\, 1 \le i \le N-1\,, \quad \frac{du_i}{dt} = \frac{a}{\Delta x^2}(u_{i-1}(t) - 2u_i(t) + u_{i+1}(t)) + f(x_i,t)$$

Let $U(t)$ be the vector $(u_1(t),\dots,u_{N-1}(t))$, we obtain the linear ODE system:

$$\dot{U}(t) = AU(t) + b(t)$$

where A is the tridiagonal matrix and $b(t)$ is the vector:

$$A = \frac{a}{\Delta x^2}\begin{pmatrix} -2 & 1 & & & \\ 1 & & & & \\ & & \ddots & & \\ & & & & 1 \\ & & & 1 & -2 \end{pmatrix} \quad b(t) = \begin{pmatrix} \theta_0(t) + f(x_1,t) \\ f(x_2,t) \\ \vdots \\ f(x_{N-2},t) \\ \theta_1(t) + f(x_{N-1},t) \end{pmatrix}$$

Of couse as long as we want to use a fine discretization (small Δx) we are turned to the integration of high dimensional systems. And that is exactly the key point! This method is known as the line method or the semi-discretization method. We will describe such a theory in another paper.

The previous example is very important : the philosophy behind this is that numerical integration of PDE could be brought back to the integration of systems of ODE (in general huge systems of nonlinear ODE). Then, in general, numerical analysis tries to solve this system by a time discretization of the ODE.

The basic idea is the same for integration of differential equations : can we identify classes of dynamical systems that could be completely studied from an *algorithmic* point of view, that-is-to-say where we can give a complete algorithmic description of the phase portrait. Of course we know the linear systems $X' = AX$ and this is a very important class of examples. We also know the linear DEA $AX' = BX + u$ and it's also fundamental.

From several places (computer science, mechanics, biology) appear other classes of powerful systems very useful for the modelling process. These systems are called cellular networks, self-reproducing automata, hybrid systems, timed automata, iterated functions systems, ... In the second part of this paper we try to give a kind of classification (at least the state of our knowledge on this large subject ...) and we identify some very deep key points. We called this part the static part because the overall structure is fixed (in a very precise sense).

Now we have to explain our **second key point**.
All these systems are useful in two ways:

- The first point is that these systems are useful as modelling tools. The example of the Glass systems is striking. The boolean networks are also very important objects of study. It's important for the future to have a complete knowledge of such systems in the same manner as we want to study the elementary functions. The phase portraits of such systems are the equivalent of the table of special functions.

- The second point is that the phase portraits of such systems could be used as **pieces** of piecewise systems (subset of the class of hybrid systems) like pieces of polynomials could be seen as pieces of spline functions.

A huge amount of work has to be done for a clear understanding of the theory! But we explore in the third part another point of view.

The **third key point** is that the dynamical systems are, in a certain sense, static. The evolution is described in a fixed phase space. But a lot of examples (see part three) show that this phase space could also be subject to evolution. We introduce the concept of dynamical automata in order to explain our point of view.

2 Dynamical and hybrid systems

2.1 Networks of automata over a graph

The first class of dynamical systems we propose is the class of networks of automata over a graph, or *automata graphs*.
Let $\mathcal{G} = (V, E)$ be an oriented graph. We note, for each vertex $v \in V$, $d^-(v)$ the inner degree of v, i.e.:

$$d^-(v) = \#\{u \in V\,,\ (u, v) \in E\}$$

If V is infinite, we say that $\mathcal{G}$ has a bounded indegree if for all $v \in V$, $d^-(v)$ is finite. Here is a common definition of automata graphs:

Definition 2.1. An automata graph is a quartet $G = (\mathcal{G}, \mathcal{S}, \mathcal{N}, \delta)$ where:

- $\mathcal{G}$ is an oriented graph with K-bounded indegree. $\mathcal{G} = (V, E)$. The elements of V are called *processors, automata, ...*
- $\mathcal{S} = \{S_v\,,\ v \in V\}$ is a collection of finite sets. The set S_v is called *state set* of automaton v.
- $\mathcal{N} : V \to (\{1, \ldots, K\} \to V)$ determines the *neighborhood* of each automaton. $\mathcal{N}(v)(\{1, \ldots, d^-(v)\})$ is in fact the *ordered* set of the predecessors of automaton v.
- δ is the *local transition function* of the network: $\delta = \{\delta_v\,,\ v \in V\}$ with

$$\delta_v : \left(\prod_{i=1}^{d^-(v)} S_{\mathcal{N}(v)(i)} \right) \to S_v$$

The set $\mathcal{Q} = \prod_{v \in V} S_v$ is called the configuration space of G.
A configuration of G is a function $c : V \to \mathcal{Q}$.

This definition is general enough to include different formalisms such as Hopfield neural networks or cellular automata. Actually it includes every discrete dynamical system that can be brought back to iterations of a map over a countable set (see [4]). To associate to an automata graph a dynamical system, we need to define an *operating mode*, that-is-to-say a strategy that determines, at each discrete time $n \in \mathbb{N}$ the subset $\Psi(n)$ of V that contains the automata that computes, thanks to the activation function δ, their next state from the states of their neighbors.

More precisely, if the system is in a configuration c_n at time n, then the next configuration c_{n+1} is given by:

$$\begin{cases} c_{n+1}(v) = c_n(v) & \text{if } v \notin \Psi(n) \\ c_{n+1}(v) = \delta_v\left(c_n(\mathcal{N}(v)(1)), \ldots, c_n(\mathcal{N}(v)(d^-(v)))\right) & \text{if } v \in \Psi(n) \end{cases}$$

The trajectories of such systems (i.e. the sequences of configurations) are the solutions of iterations of a map over the configuration space $\mathcal{Q}$:

$$\begin{cases} c_0 \in \mathcal{Q} \\ c_{n+1} = F_n(c_n) \end{cases}$$

where $F_n : \mathcal{Q} \to \mathcal{Q}$. If V is finite, then $\mathcal{Q}$ is finite and the phase portrait of the system can be represented by a finite graph $\mathcal{GT}$ which vertices are the elements of $\mathcal{Q}$. This graph is called *transition graph* of the system.

The analysis of such graphs is very hard in the general case. First, the cardinality of the phase space $\mathcal{Q}$ is exponential with respect to the number of automata. An exhaustive description of $\mathcal{GT}$ for high dimensional systems is therefore impossible.

A second difficulty of these systems lies in the choice of operating mode Ψ. If no specific assumption is made, a configuration $c \in \mathcal{Q}$ may have different successors at different instants (that is why the mapping F_n depends of the time variable n in the previous expression). This implies that the transition graph is nondeterministic, which make its analysis harder. In most of the cases, some assumptions are made on Ψ to ensure that the mapping F_n is independent of n (we then note it by F), i.e. that $\mathcal{GT}$ is deterministic. Among these assumptions, the most frequent is to consider a synchronous operating mode, in which every automata evolve at each n (see [4] for other examples).

An interesting instance of these system is the case of boolean networks. In this case, we assume that V is finite ($V = \{1, \ldots, N\}$) and that the state sets of each atomaton is the field with two elements $\mathbb{F}_2 = \{0, 1\}$. In that case any boolean function F can be seen as a multi-valuate polynomial over $\mathbb{F}_2^N$. Boolean networks have been extensively studied as they provide models of biological regulation networks. Some interesting analysis of the phase portraits of these systems have been done. We will cite among other examples the study of the links between topology and dynamics of the networks [6] and the issue of the identification of networks [4].

2.2 Glass systems

The second class of systems we propose also comes from biology. In the Glass model (see [3]), we consider the evolution of a continuous variable $X = (x_1, \ldots, x_N)$ lying in a rectangular domain of $\mathbb{R}^N$:

$$D = \prod_{i=1}^{N} [0, M_i]$$

where M_i are positive constants. We suppose that this domain is subdivided into rectangular boxes thanks to the existence of different thresholds in each directions:

$$\forall 1 \leq i \leq N\,,\ 0 = \theta_{i,0} < \theta_{i,1} < \cdots < \theta_{i,q_i} = M_i$$

To each $a = (a_1, \ldots, a_N) \in \prod_{i=1}^{N} \{1, \ldots, q_i\}$ we associate a box B_a:

$$B_a = \prod_{i=1}^{N} [\theta_{i,a_i-1}, \theta_{i,a_i}]$$

The definition of a Glass system is given by the ODE system:

$$\dot{X} = \Gamma(X) - \Lambda X \tag{1}$$

where Λ is a diagonal matrix (decay rates) and Γ is a function $D \to \mathbb{R}_+^N$ (production rates) that is constant on every boxes B_a of D. So this system is piecewise linear on D. Actually, on every box, the system is a very simple diagonal linear system whose resolution can be made explicitely.

Given N integers $q_1, \ldots, q_N$, let us note $\mathbb{N}_{q_i}$ the set $\{0, 1, \ldots, q_i\}$ and $\mathbb{D} = \prod_{i=1}^{N} \mathbb{N}_{q_i}$. Then there exists a bijection between $\mathbb{D}$ and the set of rectangular boxes of D. Given a trajectory $x(t) \in D^{\mathbb{R}}$ in the phase portrait of the system, we can consider the discrete sequence of boxes of this trajectory that consists in a trajectory of a discrete dynamical system over $\mathbb{D}$. We fall then in the previous case with an asynchronous operating mode, which makes the analysis quite hard (nondeterministic transition graph).

In order to do this analysis, an idea is to consider a partitionning of the boxes into different pieces, each piece having a unique successor. In [3], first steps in that direction are made, using tools of symbolic dynamic theory.

Remark 2.2. In order to mention this, we used the notion of trajectories of system (1) over D. Let us note that the notion of solution of system (1) is not simple! Actually, the system is well defined on opens $\mathring{B}_a$, however to consider properly solutions over the whole domain we have to use the notion of solution in the sense of Filippov [7]. We won't go further here in that direction.

2.3 Hybrid systems

The last class of systems we would like to evoke is the class of *hybrid systems*. This is a very wide class of systems that embed automata graphs and Glass systems. For a complete definition of a hybrid system see [1, 2, 5]. As an analysis of this definition is too long to be written in this paper, we will refer to these thesis for more details.

What we want to focus on here is the already mentionned notion of hybrid computation. Consider a nonlinear autonomous dynamical system given by the system of ODE:

$$\dot{X} = f(X(t)) \tag{2}$$

on an open Ω of $\mathbb{R}^n$. The direct analysis of its phase portrait is strongly dependent on the shape of the vector field f. In order to have an algorithmic way to tackle this problem, we consider a partition of Ω such that on each element of this partition we can replace the system of ODE by a more simple one, i.e. a system for which the analysis of the phase portrait can be done easily.

For instance, an important class of systems that are well known are linear systems. A first step is therefore to linearize system (2). This gives good results but only locally on several points of the phase space. In order to have a global approach, a second idea is to consider a simplicial partitionning of Ω: $(D_i)_{1\leq i\leq p}$. On each simplex D_i, we make a linear approximation of f (for instance using interpolation at the vertices of D_i). Then we have to deal with a collection of linear systems:

$$\dot{X}_i = A_i X_i(t) + b_i \ , \quad X_i(t) \in D_i$$

where A_i is a $n \times n$ matrix and b_i a n dimensional vector (see [2] for the details of the construction of these systems). Therefore we obtain a hybrid system with p discrete states and p linear dynamical systems. We have then some convergence results [2] that allow us to link the trajectories of this hybrid system to the initial one.

This technique has been applied on several examples [1, 2, 5], including examples coming from the modelling of biological systems [4]

Remark 2.3. We presented a specific subclass of hybrid systems which is the class of piecewise linear dynamical systems. Linear differential equation are used because of their simplicity and the knowledge we have on them. However, another class of differential equations have come to our attention, it is a set of equations based on power-laws known as S-systems [4]. These equations, often used in biological applications seem to have some interesting properties. An attempt to consider piecewise S-systems is currently in progress.

Remark 2.4. Again, we used in this part the notion of solution of hybrid systems which is a quite complex notion. We refer to [1] for some reflexions about *hybrid* solutions.

3 Generalized hybrid systems

Dynamical graphs [8] are very useful to model communication networks [9], embedded systems or biological behaviors. For instance, collaboration or communication between ants, behavor of a set of cells which share some local information. It is a really new type of problem and very recent models have been developped [10]. The difficulty of such structures is to deal with evolution of number of cells (or nodes), connections, states of cells, and eventually, states of edges, at the same time.

Lindenmayer has proposed a model for linear structure [11] called the L-systems.

3.1 L-systems

The simplest example of L-systems is a context-free system which is called an OL-system. Let define an alphabet Σ. Σ^* denotes a set of words over Σ and Σ^+ the set of nonempty words over Σ. The string OL-system is an ordered triple $G = \{\Sigma, w, P\}$ with $w \in \Sigma^+$ a non empty word called the axiom and $P \subset \Sigma^+$ the set of rules productions. The rules of productions are applied simultaneously. This synchronicity is the most important difference between the L-system and the classical Chomsky grammars. This mechanism is motivated by cell divisions in multicellular but linear organisms.

More general L-systems have been developped :

- Parametric L-systems[13]

  The words on which the system operate are parametric words, they can be seen as a file of cells. Each cell is built thanks to

  – a letter belonging to Σ, and we can interpret this letter as the state of the cell.
  – several parameters that can be seen as the relevant variables for the description of the evolution of the cell (time, concentration...) So a cell is represented by a module of the form

  $$q(x_1, ..., x_{n_q}) \ where \ q \in \Sigma \ and \ \{x_1, ..., x_{n_q}\} \ is \ a \ set \ of \ parameters$$

- dL-systems [14]

  A dL-system is defined as a parametric system using parametrized words $q(x_1, ..., x_{n_q})$. A module $q(x_1, ..., x_{n_q})$ is in the state q and he will stay in this state as long as the parameters will stay inside some domain $\Omega_q \subset \mathbb{R}^{n_q}$. As long as he is in Ω_q the module stay in state q but his parameters evolve.

  The evolution of these parameters is governed by a differential equation

  $$\dot{X} = f_q(X)$$

  where $X = (x_1, ..., x_{n_q})$ and f_q is some vector field on Ω_q.

3.2 Hybrid systems based on dynamical graphs

We propose a more general model which can be considered as a generalized hybrid system.

A dynamic graph is a graph where the number of nodes and the connections can change.

Let $G = (V(G), E(G))$ a graph. $V(G)$ is the set of vertices of G and $E(G)$ the set of edges. The cardinal of $V(G)$ is denoted $n(G)$. The adjacency matrix of G is denoted C_G. $C_G \in \mathcal{M}_{n(G)}$ where $\mathcal{M}_{n(G)}$ is the set of all boolean square matrix of dimension $n(G)$.

Let us suppose that the nodes of G will have values belonging to some set $\mathcal{E}$ ($\mathcal{E}$ could be a finite set, a finite field, $\mathbb{R}$, ...)

$S(G)$ is the state space of the system and is equal to $S(G) = \mathcal{E}^{n(G)}$. A dynamic on G will be described by a flow

$$\Phi_G : \mathbb{R} \times \mathbb{R} \times S(G) \to S(G)$$
$$(t, t_0, X^G(t_0)) \to X^G(t)$$

(Here we assume that time belongs to $\mathbb{R}$. If the time belongs to $\mathbb{Z}$, Φ_G could be associated to a transition function $\delta_G(X^G(0)) = \Phi(1, 0, X^G(0)))$

Now, in order to speak about generalized hybrid systems, we have to introduce some subsets of $S(G)$ where we can applied Φ_G :

$\mathcal{D}(G, \Phi_G) \subset S(G)$ is the subset where Φ_G is the legal dynamic. Of course $X^G(t_0)$ has to belong to $\mathcal{D}(G)$.

This part of the hybrid dynamic will be defined as follow :

- While $X^G(t) \in \mathcal{D}(G, \Phi_G)$ apply Φ_G
- If $X^G(t_1) \notin \mathcal{D}(G, \Phi_G)$ then apply a discrete transition.

A discrete transition arises from a set of rules. It depends on the exiting point of the domain.

We denote $\mathcal{F}$ the set of flows and we denote $\mathcal{P}(\mathcal{E}^\infty)$ the set of parts of $\mathcal{E}^\infty$. ($\mathcal{E}^\infty = \cup_{n\in\mathbb{N}}\mathcal{E}^n$.)

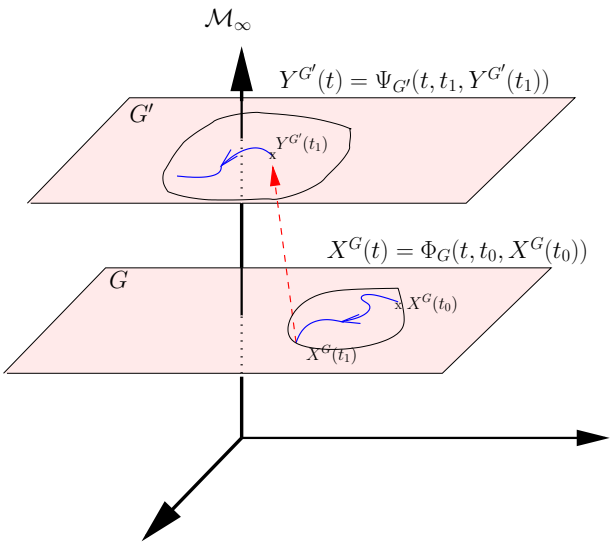
The set of rules is defined by :

$$P : \mathcal{M}_\infty \times \mathcal{E}^\infty \times \mathcal{F} \times \mathcal{P}(\mathcal{E}^\infty) \to \mathcal{M}_\infty \times \mathcal{E}^\infty \times \mathcal{F} \times \mathcal{P}(\mathcal{E}^\infty)$$

$$\left(C_G, X^G(t^*), \Phi_G, \mathcal{D}(G, \Phi_G)\right) \to \left(C_{G'}, Y^{G'}(t^*), \Psi_{G'}, \mathcal{D}(G', \Psi_{G'})\right)$$

- If $G = G'$, P is a classical hybrid reset. $C_G = C_{G'}$ and for $t > t^*$, $Y^G(t) = \Psi_G(t, t^*, Y^G(t^*))$ is the new dynamics of the system in the domain $\mathcal{D}(G, \Psi_G)$.
- If $G \neq G'$, the transition induces a dynamic of the graph. The new graph G' is defined by its adjacency matrix $C_{G'}$. It can have a new size and connections can have changed. A new dynamics $Y^{G'}(t) = \Psi_{G'}(t, t^*, Y^{G'}(t^*))$ for all nodes is defined in the domain $\mathcal{D}(G', \Psi_{G'})$

This model is explained in the following figure :



3.3 Example

Let us consider a set of cells V. This set has a set of connections defined by E. This structure is a graph $G = (V, E)$. $card(V) = n$. The state of a cell i is defined by the parameter X_i. X_i is growing depending on time :

$$X_i(t) = (\Phi_G)_i = (2 - \frac{\nu_i}{2})(t - t_0) + X_i(t_0)$$

ν_i is the number of connections of the cell i.

We consider a constraint of reproduction : If $X_i = 10$ then a new cell connected to i is added. Its index is $n+1$, $X_i = 8$ and $X_{n+1} = 2$. $\mathcal{D}_i(G, \Phi(G)) =]-\infty, 10[$.

The discret transition leads to a new graph G'. If at time t^* a transition occurs thanks to the node i, the new adjacency matrix is

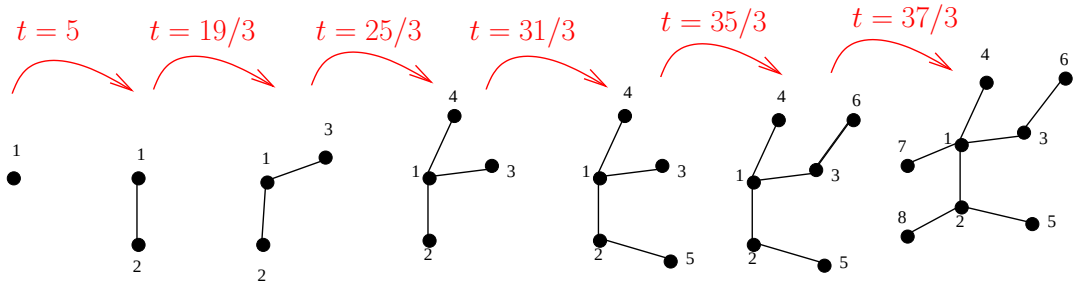
$$C_{G'} = \begin{pmatrix} \boxed{\begin{matrix} c_{11}... & c_{1i} & ...c_{1n} \\ . & . & . \\ c_{i1}... & c_{ii} & ...c_{in} \\ . & . & . \\ c_{n1}... & c_{ni} & ...c_{nn} \end{matrix}} & \boxed{\begin{matrix} 0 \\ . \\ 1 \\ . \\ 0 \end{matrix}} \\ \boxed{0 \ ... \ 1 \ ... \ 0} & 0 \end{pmatrix}$$

where $C_G = (c_{ij})$ was the previous adjacency matrix.

We easily extend this rule if more than one cell reach the boundaries of $\mathcal{D}(G, \Phi(G))$ at the same time.

At time $t = 0$, we suppose there is only one cell, the state of the system is $(C_G, X_0^G, \Phi_G) = ((0), (0), (2t))$. The structure of the graph is stable until $t = 5$. At time $t = 5$ a discret transition is perfomed : a new cell is added, the state of the system becomes $(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 8 \\ 2 \end{pmatrix}, \begin{pmatrix} \frac{3}{2}t + \frac{1}{2} \\ \frac{3}{2}t - \frac{11}{2} \end{pmatrix})$.

The following figure shows how the graph evolves and how Φ_G evolves.



$$(2t)\begin{pmatrix} \frac{3}{2}t + \frac{1}{2} \\ \frac{3}{2}t - \frac{11}{2} \end{pmatrix}\begin{pmatrix} t + \frac{5}{3} \\ \frac{3}{2}t - \frac{11}{2} \\ \frac{3}{2}t - \frac{15}{2} \end{pmatrix}\begin{pmatrix} \frac{t}{2} + \frac{23}{6} \\ \frac{3}{2}t - \frac{11}{2} \\ \frac{3}{2}t - \frac{15}{2} \\ \frac{3}{2}t - \frac{21}{2} \end{pmatrix}\begin{pmatrix} \frac{t}{2} + \frac{23}{6} \\ t - \frac{7}{3} \\ \frac{3}{2}t - \frac{15}{2} \\ \frac{3}{2}t - \frac{21}{2} \\ \frac{3}{2}t - \frac{27}{2} \end{pmatrix}\begin{pmatrix} \frac{1}{2}t + \frac{23}{6} \\ t - \frac{7}{3} \\ t - \frac{11}{3} \\ \frac{3}{2}t - \frac{21}{2} \\ \frac{3}{2}t - \frac{27}{2} \\ \frac{3}{2}t - \frac{31}{2} \end{pmatrix}\begin{pmatrix} 8 \\ \frac{1}{2}t + \frac{11}{6} \\ t - \frac{11}{3} \\ \frac{3}{2}t - \frac{21}{2} \\ \frac{3}{2}t - \frac{27}{2} \\ \frac{3}{2}t - \frac{31}{2} \\ \frac{3}{2}t - \frac{33}{2} \\ \frac{3}{2}t - \frac{33}{2} \end{pmatrix}$$

The evolution of the system is completly asynchronous. The transition times (defined by a completly deterministic process) defined by a sequence $\{t_i\}_i$ is a part of the description of the flow. The question is how to describe properly and precisely the evolution of such examples.

The state space of the flow of the system is the state flow of the union of all the state flows of the existing nodes. Moreover we have to consider the discrete evolution of the graph during time. Important information for a graph are picked up from the adjacency matrix : the number of nodes, the maximal degree of the set of nodes, the average of the degree of all nodes, the proportion of nodes of constant degrees, numbers and length of cycles and clicks. and also the spectrum of the laplacian of the adjacency matrix [15].

References

[1] M. Mirica-Ruse, *Contribution à l'étude des systèmes hybrides.* Thèse de Doctorat, UJF Grenoble, 2002.

[2] A. Girard, *Analyse algorithmique des systèmes hybrides.* Thèse de Doctorat, INP Grenoble, 2004.

[3] E. Farcot, *Etude d'une classe d'équations différentielles affines par morceaux modélisant des réseaux de régulation biologique.* Thèse de Doctorat, INP Grenoble, 2005.

[4] L. Tournier, *Etude et modélisation mathématique de réseaux de régulation génétique et métabolique.* Thèse de Doctorat, INP Grenoble, 2005.

[5] A. Rondepierre, *Algorithmes hybrides pour le contrôle optimal des systèmes non linéaires.* Thèse de Doctorat, INP Grenoble, 2006.

[6] J.J. Fox and C.C. Hill, *From topology to dynamics in biochemical networks.* Chaos, 11(4):809–815, 2001.

[7] A.F. Filippov, *Differential equations with discontinuous righthand sides.* Mathematics and their Applications (Soviet series), vol. 18, Kluwer Academic, Dordrecht, 1988.

[8] F. Harary, G. Gupta, *Dynamic Graph Models.* Math. Comput. Modelling Vol. 25, No. 7, p 79-87, 1997.

[9] S. Bohacek, J. Hespanha, J. Lee, K. Obraczka, *A Hybrid Systems Modeling Framework for Fast and Accurate Simulation of Data Communication Networks.* SIGMETRICS'03, June 10-14, 2003, San Diego, California, USA.

[10] F. Kratz, O. Sokolsky, G. Pappas, I. Lee, *R-Charon, a Modeling Language for Reconfigurable Hybrid systems.* accepted to HSCC 2006.

[11] A Lindenmayer, *Mathematical models for cellular interaction in development (Parts I and II).* Journal of Theoretical Biology, 18:280-315, 1968

[12] P Prusinkiewicz, A Lindenmayer, *The algorithmic beauty of plants.* Springer-Verlag

[13] P Prusinkiewicz, J Hanan, *Visualization of botanic structures and processes using parametric L-systems.* In D. thalmann editor, Scientific visualization and Graphics Simulations. pages 183-201, J. Wiley & Sons, 1990.

[14] P Prusinkiewicz, Marc Hamel, Eric Mjolsness, *Animation of Plant Development.* Proceedings of SIGGRAPH 93 (Anaheim, California, August 1-6) In Computer graphics Proceedings, Annual Conference Series, 1993, ACM SIGGRAPH, pp 351-360.

[15] Y. Kim, M. Mesbahi, *On Maximizing the Second Smallest Eigenvalue of a State-dependent Graph Laplacian.* submitted to 2005 American control Conference.

[16] Eugene Asarin, Paul Caspi, O. Maler, *Timed regular expressions.* Journal of the ACM 49, No.2, 2002, 172-206.

Jean Della Dora
Laboratoire de Modélisation et Calcul (LMC-IMAG)
Institut National Polytechnique de Grenoble
Jean.Della-dora@imag.fr

Aude Maignan
Laboratoire de Modélisation et Calcul (LMC-IMAG)
Université Pierre Mendès France
Aude.Maignan@imag.fr

Laurent Tournier
Laboratoire de Modélisation et Calcul (LMC-IMAG)
Université Pierre Mendès France
Laurent.Tournier@imag.fr

Does Church-Turing thesis apply outside computer science?

Eugene Asarin

Abstract

We analyze whether Church-Turing thesis can be applied to mathematical and physical systems. We find the factors that allow to a class of systems to reach a Turing or a super-Turing computational power. We illustrate our general statements by some more concrete theorems on hybrid and stochastic systems.

> The future of mathematics comes from informatics, the future of informatics comes from mathematics
>
> *Thoughts,* Jean Della Dora

1 Introduction

The discovery in the first half of the XX-th century of several computational models (such as Post and Turing machines, recursive functions, Markov's normal algorithms etc.), all capable to realize any imaginable algorithm, and the proof of their equivalence, can be seen as the first step of the computer science. They determined the invention of computers, design of programming languages, and the development of a new kind of mathematics that evolved to theoretical computer science. All the above-mentioned computational models are based on unbounded discrete-time, unbounded discrete-state memory, deterministic programs, and no-noise no-faults execution. Such restrictions were a key to the correct (and unique) definition of computability, as well as to the digital computers, based on special engineering tricks allowing to represent discrete state, discrete time, and noise rejection in the physical world.

According to Church-Turing thesis, any reasonable (i.e. satisfying above-mentioned restrictions) computational model leads to the same (or smaller) class of computable functions as Turing machines, and hence Turing machines capture the general notion of algorithm.

However a challenging research direction consists in considering any natural class of dynamical systems (possibly continuous in time and/or state-space, possibly non-deterministic, possibly noisy, possibly quantic) as a computational model, and exploring its computational power. It could be interesting for several reasons. The most important one is to verify or to falsify the following thesis.

Thesis 1.1 (Extended Church-Turing). *Feasible, realizable, reproducible physical computing devices have the same (or smaller) computational power as Turing machines.*

In other words, according to this thesis, there is no way to build a computer, based on some new principles and capable to "compute" other functions than Turing computable ones. As far as I know, there is no scientific evidence in favor of this thesis, but almost everybody believes in it. Since it is difficult to define rigorously what a feasible *physical* device is, a possible approach to this thesis could be to consider various classes of *mathematical* models, to analyze their computational power, and whenever it exceeds the power of Turing machines, to think about its physical feasibility.

Another scientific application of the computability approach to dynamical systems is to use the computational power as a natural measure of behavioral complexity. The bigger is the set of functions computed by a class of dynamical systems, the more complex, strange, pathologic can be the behavior of such systems. In fact complexity hierarchies of the theoretical computer science (similar to forgotten hierarchies of the descriptive set theory, but easier to study) are quite rich, and can shed new light on chaos and other strange attractors. A very high level of complexity can be interpreted as a sign of unrealizability of the system.

Last but not least, practically speaking the computational approach to dynamical systems is a key to the decidability analysis of verification problems concerning hybrid and continuous systems.

The computability view of dynamical systems is not really new, it is related to research on computability on reals[12], super-Turing models, hybrid systems, cellular automata and so on.

In this talk, several concrete research works on various aspects of computability by dynamical systems will be presented. The author was involved in these works during last 12 years, see [1]-[5].

2 Continuous systems as computational models

Most of results will be established for a class of piecewise-constant differential equations (so-called PCD systems) that appears in Hybrid systems research, and its variants. Computability by such devices will be defined, and their computational power will be explored in following situations.

Dynamical systems weaker then TMs : Poincaré - Bendixson's paradise[1, 4]. It is well-known that differential equations on the plane have rather simple global behavior. In the computational paradigm this leads a weak (sub-Turing) computational power of planar systems.

Dynamical systems as strong as TMs : chaos[1]. Starting from three dimensions dynamical systems admit chaotical behaviors. We will analyze how the chaos allows to match the full Turing computational power (see also [10]).

Far beyond Turing : Zeno's horror[2, 6]. The so-called Zeno phenomenon for piecewise continuous (and even piecewise constant) systems consists in the simple fact that a trajectory can have infinitely many changes of the type of dynamics during a finite interval of time. Such a behavior is much more complex than the simple deterministic chaos. This kind of extremely complex trajectories can be used to decide any arithmetic predicate in a bounded lapse of time. Hence the computational power of Zeno dynamical systems goes far beyond Turing machines. We will speculate about physical (non)-realizability of Zeno computers because of their extreme sensibility to perturbations.

Small set-valued noise: upside down TMs[3]. Several researchers (in particular Henzinger, Raskin [9] and Fränzle[7]) suggested to consider more realistic, imprecise, perturbed dynamical systems. A folk conjecture suggests that adding a small imprecision to the dynamics makes the reachability problem decidable, by destroying too subtle behaviors, and replacing them by thick "tubes" that rapidly cover parts of the state-space. However, up to now there are no convincing decidability results in this direction.

Inspired by Puri's work [11] we have tried another approach: replace the exact dynamics of a system by an ϵ-perturbed contingent one with $\epsilon \to 0$. As the result the computational power becomes Π_1^0. This means that instead of recognizing recursively enumerable sets, "perturbed" systems recognize complements of such sets. We will explain the reason of this inversion in computational power.

Large deviations or small stochastic noise: one step beyond Turing[5]. Another one, maybe more realistic model takes into account a small stochastic noise, and to pass to the limit as it is often done in the large deviations research (see [8]). A characterization of the computational power of such perturbed systems will be presented. It turns out to be Δ_2^0, i.e. slightly superior to Turing machines. We will relate it with the complexity of behaviors appearing as large deviations of deterministic dynamics.

3 Conclusions

We believe that the computability approach to dynamical systems can be a source of new insights and new research problems. We will present some important open problems that exist in this area.

Acknowledgments. This talk is based on joint work with: Ahmed Bouajjani, Pieter Collins, Oded Maler, Amir Pnueli, Gerardo Schneider and Sergio Yovine and valuable discussions with Jean Della Dora, Vincent Blondel, Olivier Bournez, Martin Fränzle, Jean-François Raskin and many other colleagues.

References

[1] E. Asarin, O. Maler, A. Pnueli, Reachability analysis of dynamical systems having piecewise-constant derivatives, Theoretical Computer Science 138, 35-65, 1995.

[2] E. Asarin, O. Maler, Achilles and the Tortoise Climbing Up the Arithmetical Hierarchy, J. of Computer and System Sciences 57, 389-398, 1998.

[3] E. Asarin, A. Bouajjani. Perturbed Turing machines and hybrid systems. In: Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society (2001) 269-278.

[4] E. Asarin, G. Schneider, S.Yovine. On the Decidability of the Reachability Problem for Planar Differential Inclusions. In Hybrid Systems: Computation and Control, 89-104, LNCS 2034, 2001.

[5] E. Asarin, P. Collins, Noisy Turing Machines, in. Proc. of ICALP'05, 1031-1042 LNCS 3580, 2005.

[6] O. Bournez. Achilles and the Tortoise Climbing Up the Hyper-arithmetical Hiearchy. Theoretical Computer Science, 210 (1): 21-71, 1999.

[7] M. Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In : Computer Science Logic, 126–140, LNCS 1683, 1999.

[8] M. Freidlin, A. Wentzell. Random perturbations of dynamical systems. Springer-Verlag, New York, 1984.

[9] Th.A. Henzinger, J.-F. Raskin. Robust undecidability of timed and hybrid systems. Hybrid Systems: Computation and Control, 145-159, LNCS 1790, 2000.

[10] C. Moore, Generalized shifts: unpredictability and undecidability in dynamical systems, Nonlinearity 4 (1991), 199-230.

[11] A. Puri. Dynamical properties of timed automata. Discrete Event Dynamic Systems **10** (2000) 87–113.

[12] K. Weihrauch. Computable analysis. Springer-Verlag, Berlin, 2000.

Eugene Asarin
LIAFA
University Paris 7 & CNRS
asarin@liafa.jussieu.fr
www.liafa.jussieu.fr/~asarin

A Hidden Algebraic Structure in Quantum Mechanics

Léon Brenig

Abstract

Quantum mechanics of a non-relativistic free particle implies the existence of a second time variable associated to a non-unitary transformation group. This group is part of a non-unitary representation of the Galilei group. The unitary and non-unitary algebras are connected via a group of Lorentz-like transformations corresponding to measurement accuracy transformations preserving the Heisenberg inequality. Quantum mechanics appears in this context as a relativity theory of precision in measurements.

1 Introduction

The relation between quantum and classical mechanics is a fundamental question that aroused many interesting works. From Weyl geometry [1] [2] to fractal space-time or scale-relativity [3] and from the Bohm interpretation [4] to the stochastic approach [5] [6], important results in theoretical and mathematical physics have been achieved. An excellent review of these progresses along with an exhaustive bibliography on the subject can be found in [7] [8]. However, important conceptual difficulties still remain concerning the transition between classical and quantum mechanics. The latter, in contrast with classical mechanics, does not appear as a closed theory. It needs classical mechanics through the correspondance principle in order to construct the hermitian operators that represent the observable physical quantities. Furthermore, quantum mechanics involves two types of processes as emphasized by R.Penrose [9], the unitary evolution of the system's state vector and the reduction or collapse of the latter. The former corresponds to the evolution of the system in absence of any measurement made by an observer. The second represents the process that occurs when an external measurement instrument is applied to the system. The measurement process produces a sudden interruption of the unitary evolution of the state vector. The latter is instantaneously projected on an eigenvector of the observable that is measured. Up to now, quantum mechanics has no explanation for this category of processes. More generally, there exist in quantum mechanics two other types of processes having in common with the measurement process the strange property to be instantaneous. These are the transitions between quantum energy levels in atoms and the instantaneous transfer of information related to experiments handling with non-locality. Quantum mechanics does not solve the physical paradox of these infinitely rapid transformations, though, one can show that they do not violate the relativity principle.

We show in this work that the "unfinished" aspect of todays quantum mechanics is related to the existence of a hidden algebraic structure that associates to the known unitary group

generated by the hermitian observables a non-unitary group of transformations that was hidden up to now. This structure implies the existence of a second time dimension. Furthermore, a continuous group of transformations acting on the precision of the measurement is involved in this algebraic structure. With this result, quantum mechanics appears as a relativity of precision theory. This new perspective is a step toward a deeper reconciliation between quantum mechanics and special relativity. The latter emphasizes the role of the observer and relies on the invariance of the laws of physics for the class of inertial observers. In this class, the observers are characterized by their relative velocity. In contrast, quantum mechanics in its current state does not attribute a specific role to the observer. This concept is evacuated in quantum physics. The group of precision transformations we have found in the present work acts on a new characteristics of the observer, the precision of its instruments. We show that quantum mechanics corresponds to a postulate of invariance of physical laws under these precision transformations between observers.
To unveil this algebraic structure we shall not work in the usual algebraic framework of quantum mechanics, that is, the group of unitary transformations generated by the algebra of hermitian operators on the Hilbert space. We use, instead, a functional canonical representation of this algebra, as we show in the next chapter.
To give a foretaste of this representation, let me give a representative example of a member of that functional canonical algebra. This is the Hamiltonian functional, i.e. the quantum average of the energy for a non-relativistic free particle, expressed in terms of the probability density, ρ(x), and the action, s(x) by:

$$\mathcal{H} = \int d^3\, x \;\; [\frac{\rho(x)\left|\nabla s(x)\right|^2}{2m} + \frac{\hbar^2}{2m}\left|\nabla \rho(x)^{1/2}\right|^2\Big] \tag{1}$$

where the integral is over $\mathbb{R}^3$, ∇ is the tridimensional gradient, $\hbar^2$ is the Planck constant divided by 2π and m is the mass of the particle. The density, ρ(x), denotes the probability density of finding the particle at the space point x. $\mathcal{H}$ is a functional of ρ(x) and s(x).
In the expression (1) the second term in the integrand corresponds after some simple calculation to the contribution of the so-called quantum potential [4], $-\frac{\hbar^2}{2m}\frac{\nabla^2\rho^{1/2}}{\rho^{1/2}}$,while the first term represents the kinetic energy density in classical mechanics. This second term represents the basic difference between quantum and classical mechanics.
The hamiltonian functional $\mathcal{H}$ (1) can also be considered as a functional of the wave function by noting that

$$\mathcal{H} = \int d^3x \;\; \psi^* h \psi \qquad \text{with} \quad h = -\frac{\hbar^2}{2m}\nabla^2 \quad \text{and} \quad \psi = \rho^{1/2} e^{\mathrm{i}s/\hbar}. \tag{2}$$

From (2), $\mathcal{H}$ is obviously the quantum average of the hamiltonian hermitian operator, h.
In the classical limit where $\frac{\hbar^2}{2m}$ tends to zero, the hamiltonian functional, $\mathcal{H}$, generates the classical evolution of a point particle of mass m with a random initial position characterized by the probability density ρ.
This article shows that the apparition of the quantum potential in equation (1) is intrinsically related to the existence of a second time variable as said above. The evolution of

the wave function in this new time variable is not unitary. The corresponding non-unitary time evolution generator together with the generators of translations, rotation and Galilean velocity changes constitutes a non-unitary representation of the Galilei algebra. There are, thus, two "mirror" Galilei algebras: The unitary one associated to the Hamiltonian (1) and the non-unitary one. Both are related by a one-parameter continuous group of transformations that, remarkably, is isomorphic to the two-dimensional Lorentz group. However, their meaning is deeply different. We show, indeed, that these transformations correspond to changes in the accuracy of physical measurements and preserve the Heisenberg inequality. In short, the picture of quantum mechanics that emerges from this twin, unitary and non-unitary, algebraic structure is that of a relativity in the precision of measurements performed by the observers.

2 The functional representation of quantum mechanics

Let me first introduce the generalized canonical formalism used in this work. It has been first described in the work of F.Guerra [6], in a context, though, very different from the problems studied here.
The functionals $\mathcal{A} = \int d^3x$ F(x, ρ, $\nabla\rho$, $\nabla\nabla\varrho$, ..., s, ∇s, $\nabla\nabla$s, ...) of ρ(x) and s(x) that are functionally differentiable with respect to the functions ρ(x) and s(x) constitute an infinite Lie algebra $\mathbb{G}$ under the functional Poisson bracket:

$$\{\mathcal{A}, \mathcal{B}\} = \int d^3\, x \;\; [\frac{\delta\mathcal{A}}{\delta\rho(x)}\frac{\delta\mathcal{B}}{\delta s(x)} - \frac{\delta\mathcal{B}}{\delta\rho(x)}\frac{\delta\mathcal{A}}{\delta s(x)}\Big] \tag{3}$$

where $\delta/\delta\rho(x)$ and $\delta/\delta s(x)$ represent the functional derivative with respect to the functions ρ(x) and s(x).
All the volume integrals appearing in this article are defined over the whole space $\mathbb{R}^3$,
In this framework, the transformation from ρ and s to the wave function ψ and its complex conjugate
ψ^*, with $\psi = \rho^{1/2}e^{\mathrm{is}/\hbar}$,can be shown to be a canonical transformation. In other words, the Poisson bracket structure (3) is transferred from the ρ, s variables to the ψ, ψ^*variables.
It has been shown by F.Guerra and R.Marra that the $\mathbb{G}$ algebra contains a sub-algebra $\mathbb{Q}$ that is isomorphic to the algebra of the quantum observable operators [6]. The elements of $\mathbb{Q}$ are functionals that are quadratic in ψ, ψ^* in the form $\mathcal{A} = \int d^3x \;\; \psi^* A\psi$ where A is a Hermitian operator. Using bracket definition (3) expressed in terms of the variables ψ, ψ^* one easily gets:

$$\{\mathcal{A}, \mathcal{B}\} = \int d^3x \;\; \psi^*[A, B]\,\psi \tag{4}$$

where $[A, B]$ is the commutator of A and B.
There is another sub-algebra, $\mathbb{P}$, in $\mathbb{G}$ that is isomorphic to the classical mechanical Poisson algebra. Its elements are of the form $\mathcal{A} = \int d^3x \;\; \rho(x)a(\mathrm{x}, \nabla s)$ where a(x, p) is any function analytic in p and where x and p are conjugate variables for the usual, non-functional,

Poisson bracket of classical mechanics. Indeed as shown in a forthcoming article [10] one has:

$$\{\mathcal{A},\mathcal{B}\} = \int d^3x \;\; \rho(x)\{a,b\}_{\rm cl} \tag{5}$$

where $\mathcal{A}$ and $\mathcal{B}$ are both arbitrary members of the sub-algebra $\mathbb{P}$ and $\{a,b\}_{\rm cl}$ is the usual classical Poisson bracket in the result of which p must be replaced by ∇s.
The time evolution of any functional $\mathcal{A}$ belonging to the algebra $\mathbb{G}$ is given by:

$$\partial_t \mathcal{A} = \{\mathcal{A},\mathcal{H}\} \tag{6}$$

where $\mathcal{H}$ is the Hamiltonian functional (1). The functions ρ(x) and s(x), themselves, are elements of $\mathbb{G}$. As such, their evolution equations are obtained by applying (6) and the rules of functional derivation:

$$\partial_t \rho = -\nabla.\left(\rho\frac{\nabla s}{m}\right) \tag{7}$$

$$\partial_t s = -\frac{|\nabla s|^2}{2m} + \frac{\hbar^2}{2m}\frac{\nabla^2\rho^{1/2}}{\rho^{1/2}} \tag{8}$$

Equation (7) is the probability conservation equation and equation (8) is a Hamilton-Jacobi equation involving the quantum potential. It is well known that the same system (7,8) is also obtained when splitting the Schrödinger equation into real and imaginary components. This reflects the equivalence between the generalized canonical time transformations generated by $\mathcal{H}$ and the quantum unitary time transformations generated by the hermitian operator H: Equation (6) applied to ψ yields the Schrödinger equation.
Let me now show, in analogy with a theory developped for non-quantum systems by E.C.G.Sudarshan [11], that a canonical functional representation of the Galilei algebra exists and is as a sub-algebra of $\mathbb{G}$. In this representation, the following functionals

$$\mathcal{P} = \int d^3x \;\; \rho(x)\nabla s(x) \tag{9}$$

$$\mathcal{L} = \int d^3x \;\; \rho(x)x \wedge \nabla s(x) \tag{10}$$

$$\mathcal{G} = m\int d^3x \;\; \rho(x)x \tag{11}$$

respectively represent the generators of translation, rotation and observer velocity change. Together with the hamiltonian functional $\mathcal{H}$ given by (1), they form a functional canonical representation of the Galilei algebra. This is shown by calculating the functional Poisson brackets (3) between those four quantities and obtaining the well-known Galilei algebra Lie bracket relations. There is a short-cut to these calculations if one remarks that these four quantities belong to the sub-algebra $\mathbb{Q}$. Hence, their brackets are in one-to-one correspondance with the commutators of the corresponding hermitian operators (see equation (4)). The latter form a unitary representation of the Galilei algebra and the proposition is, thus, proved. This representation generates a canonical representation of the Galilei group that is isomorphic to the usual unitary group of quantum mechanics.

3 The non-unitary side of quantum mechanics

We are now ready to start the proof of the main result of this article, in short: Quantum mechanics when represented in the large $\mathbb{G}$ canonical algebra appears to possess a twin group of non-unitary transformations associated to the above discussed unitary group. Both are related via a group of special transformations, isomorphic to the Lorentz transformations in 1+1 dimensions. However, in contrast with special relativity, these transformations do not relate observers with different constant velocities. In our work, these transformations appear to correspond to changes of the precision with which the observers perform their measures on the physical world.
Consider the following functional belonging to $\mathbb{G}$

$$\mathcal{S} = \int d^3x \;\; \rho(x)s(x) \tag{12}$$

It represents the average action or, up to a factor $\hbar$, the average phase of the particle. The average is taken on the ρ probability density. Note that this functional belongs to the algebra $\mathbb{G}$ but not to the quantum sub-algebra $\mathbb{Q}$. Indeed, in terms of ψ and ψ

$$\mathcal{S} = \frac{\hbar}{2i} \int d^3x \;\; \psi^* \ln(\frac{\psi}{\psi^*})\psi \tag{13}$$

which is not a quadratic form in ψ, ψ^*.
The functional Poisson bracket of $\mathcal{S}$ with $\mathcal{H}$, denoted by $\mathcal{K}$, is easily calculated yielding:

$$\mathcal{K} \equiv \{\mathcal{S}, \mathcal{H}\} = \int d^3 \; x \;\; [\frac{\rho \left|\nabla s\right|^2}{2m} - \frac{\hbar^2}{2m}\left|\nabla \rho^{1/2}\right|^2\Big] \tag{14}$$

$\mathcal{K}$ differs from $\mathcal{H}$ only by the sign of the second term in the right hand side. Its physical dimension is, thus, that of an energy. As any other element of the algebra $\mathbb{G}$, it generates a one-parameter continuous group. Since $\mathcal{K}$ is an "energy", the associated group parameter, τ, has the physical dimension of a time. Remark that $\mathcal{K}$, though belonging to $\mathbb{G}$, does belong neither to the $\mathbb{Q}$ sub-algebra nor to the $\mathbb{P}$ sub-algebra. In fact, it generates a non-unitary one parameter group.
A simple calculation brings also:

$$\{\mathcal{S}, \mathcal{K}\} = \int d^3 \; x \;\; [\frac{\rho \left|\nabla s\right|^2}{2m} + \frac{\hbar^2}{2m}\left|\nabla \rho^{1/2}\right|^2\Big] = \mathcal{H} \tag{15}$$

Any functional $\mathcal{A}$ belonging to $\mathbb{G}$ can be viewed as a function of both times t and τ, and its τ-derivative is given by:

$$\partial_\tau \mathcal{A} = \{\mathcal{A}, \mathcal{K}\} \tag{16}$$

Consider, then, the complex energy functional $\mathcal{H}$+i$\mathcal{K}$ as a function of the variables t and τ. The partial t- and τ- derivatives of the real and complex parts are:

$$\partial_t \mathcal{H} = \;\; \{\mathcal{H}, \mathcal{H}\} = 0 \text{ and } \;\; \partial_\tau \mathcal{K} = \;\; \{\mathcal{K}, \mathcal{K}\} = 0 \tag{17}$$

$$\partial_\tau \mathcal{H} = \;\; \{\mathcal{H}, \mathcal{K}\} \text{ and } \;\; \partial_t \mathcal{K} = \;\; \{\mathcal{K}, \mathcal{H}\} = -\{\mathcal{H}, \mathcal{K}\} \tag{18}$$

We, thus, obtain:

$$\partial_t \mathcal{H} = \partial_\tau \mathcal{K} \quad \text{along with} \quad \partial_\tau \mathcal{H} = -\partial_t \mathcal{K} \tag{19}$$

Equations (19) are the Cauchy-Riemann conditions corresponding to the complex analyticity of the function $\mathcal{H} + \mathrm{i}\mathcal{K}$ in the complex variable $\mathrm{t}+\mathrm{i}\tau$.

Let us now focus on the functional brackets of $\mathcal{K}$ with $\mathcal{P}$, $\mathcal{L}$ and $\mathcal{G}$. Simple calculations show that they satisfy the Galilei algebra Lie brackets. As $\mathcal{K}$ does not belong to $\mathbb{Q}$, the reasoning here cannot take the same short-cut via the $\mathbb{Q}$ algebra as in the proof that $\mathcal{H}$, $\mathcal{P}$, $\mathcal{L}$, $\mathcal{G}$ form a Galilei algebra: One must explicitely compute the functional Poisson brackets to check that they, indeed, correspond to a representation of the Galilei algebra [10]. The exponentiation of the algebra generated by $\mathcal{K}$, $\mathcal{P}$, $\mathcal{L}$ and $\mathcal{G}$ yields a representation of the Galilei group that is not unitary. This is clear from the fact that $\mathcal{K}$ does not belong to the sub-algebra $\mathbb{Q}$.

At this level of our presentation, the status of the new quantities, $\mathcal{S}$, $\mathcal{K}$ and τ must be questioned. Are these quantities physical? At first sight, they could be thought of as arbitrary mathematical objects without any link with the physical world.

A first element in favor of the relevance of these objects to physics is obtained by considering the analytic extension of the classical hamiltonian functional for a free particle

$$\mathcal{H}_{\mathrm{cl}} = \int d^3x \ \ \frac{\rho \left|\nabla s\right|^2}{2m} \tag{20}$$

into a complex function, $\mathcal{H} + \mathrm{i}\mathcal{K}$, holomorphic in the complex plane of $\mathrm{t}+\mathrm{i}\tau$. The requirements of the Galilean invariance of this analytic continuation leads to the necessity of adding a second term to $\mathcal{H}_{\mathrm{cl}}$ wich, precisely, is the quantum potential up to the value of a constant factor. The proof is rather lengthy and will be presented in [10]. However, from this result one can infer that the existence of a second time dimension is a necessary and sufficient condition for the emergence of the quantum contribution to the Hamiltonian functional. As the latter, along with the whole theoretical apparatus of quantum mechanics, leads to a wealth of experimentally verified predictions, the physical status of this second time dimension is reinforced.

However, the most decisive argument in favor of the physicality of the second time variable is related to the measurement process in quantum physics as shown in the next chapter

4 A relativity of measurement precision.

As any element of $\mathbb{G}$, the functional $\mathcal{S}$ defined in (12) generates a continuous one-parameter group of transformation. Let us call α the parameter of that group. Physically this parameter is dimensionless since $\mathcal{S}$ has the dimension of an action. Any functional $\mathcal{A}$ of $\mathbb{G}$ may be considered as a function of that parameter and obeys

$$\frac{\partial}{\partial \alpha}\mathcal{A}(\alpha) = \{\mathcal{A}(\alpha), \mathcal{S}(\alpha)\} \tag{21}$$

where $\mathcal{A}(\alpha)$ denotes $\mathcal{A}(\rho, s; \alpha)$.

The Lie algebra structure of $\mathbb{G}$ ensures that the Lie brackets relation are preserved under the transformations generated by $\mathcal{S}$. Moreover, $\mathcal{S}$ is of course an invariant under these transformations.
Let me first consider the particular case of the classical Hamiltonian functional $\mathcal{H}_{\rm cl}$ as defined in (20). One has:

$$\frac{\partial}{\partial\alpha}\mathcal{H}_{\rm cl}(\alpha) = \{\mathcal{H}_{\rm cl}(\alpha), \mathcal{S}(\alpha)\} = -\mathcal{H}_{\rm cl}(\alpha) \tag{22}$$

whose solution is:

$$\mathcal{H}_{\rm cl}(\alpha) = e^{-\alpha}\mathcal{H}_{\rm cl}(0) \tag{23}$$

where $\mathcal{H}_{\rm cl}(0)\equiv\mathcal{H}_{\rm cl}$
This transformation of the hamiltonian functional can be interpreted as a dilatation of time:

$$t(\alpha) = e^{\alpha}t \tag{24}$$

since the parameter t is associated to the group generated by $\mathcal{H}_{\rm cl}$.
Let us, now, see how the quantum time evolution generators $\mathcal{H}$ and $\mathcal{K}$ transform under the group generated by $\mathcal{S}$. We must solve the system of equations

$$\frac{\partial}{\partial\alpha}\mathcal{H}(\alpha) = \{\mathcal{H}(\alpha), \mathcal{S}(\alpha)\} = -\mathcal{K}(\alpha) \tag{25}$$

$$\frac{\partial}{\partial\alpha}\mathcal{K}(\alpha) = \{\mathcal{K}(\alpha), \mathcal{S}(\alpha)\} = -\mathcal{H}(\alpha) \tag{26}$$

whose solution is

$$\mathcal{H}(\alpha) = \cosh\alpha\mathcal{H}(0) - \sinh\alpha\mathcal{K}(0) \tag{27}$$

$$\mathcal{K}(\alpha) = -\sinh\alpha\mathcal{H}(0) + \cosh\alpha\mathcal{K}(0) \tag{28}$$

where $\mathcal{H}(0)\equiv\mathcal{H}$ and $\mathcal{K}(0)\equiv\mathcal{K}$.
These transformations may also be transferred on the time coordinates as follows

$$t(\alpha) = \cosh\alpha \;\; t + \sinh\alpha\tau \tag{29}$$

$$\tau(\alpha) = \sinh\alpha \;\; t + \cosh\alpha\tau \tag{30}$$

These are Lorentz-like transformations in the (t,τ) plane. Of course, the context in which they appear here is completely foreign to special relativity since we are considering a non-relativistic free particle as our system.
These transformations mix both types of time coordinates while keeping covariant the time evolution equations for any functional $\mathcal{A}$ of $\mathbb{G}$

$$\partial_t\mathcal{A} = \{\mathcal{A}, \mathcal{H}\} \tag{31}$$

$$\partial_\tau\mathcal{A} = \{\mathcal{A}, \mathcal{K}\} \tag{32}$$

Incidentally, under the group generated by $\mathcal{S}$, the state variables ρ and s transform as

$$\rho(\alpha) = e^{\alpha}\rho \tag{33}$$

$$s(\alpha) = e^{-\alpha}s \tag{34}$$

We now shift from the consideration of the quantities $\mathcal{H}$ and $\mathcal{K}$ to that of the statistical quantities characterizing the precision of measurements. We have in mind the quadratic deviations of the momentum and of the position of the particle. These quantities are functionals of the wave function and, as such, belong also to the algebra $\mathbb{Q}$. As is well known, these two quantities are constrained by the Heisenberg inequality. In the sequel, I show that the transformations (27) and (28) can be recasted into transformations of these deviations, and that these transformations saturate at the lower bound of the Heisenberg inequalities.

In order to prove this proposition, let me first define the above statistical quantities. The quantum average of an observable quantity represented by the associated hermitian operator A is given by:

$$\langle A\rangle = \int d^3x \;\; \psi^* A \;\; \psi \tag{35}$$

It is readily seen that the average momentum $\langle p\rangle$, for which the associated hermitian operator is p = -i$\hbar\nabla$, reduces to

$$\langle p\rangle = \int d^3x \;\; \rho(x)\nabla s(x) \tag{36}$$

or, in other terms, $\langle p\rangle = \mathcal{P}$.

The quantum quadratic deviation of p is given by

$$\left\langle |p-\mathcal{P}|^2 \right\rangle = \langle p^2\rangle - \mathcal{P}^2 \tag{37}$$

This quantity can be reduced to $\langle p^2\rangle$ by using a frame of reference traveling at the average speed of the particle, $\frac{\mathcal{P}}{m}$. This simplification represents by no means a loss of generality of the subsequent results. Indeed, a change to any other frame of reference can be performed later.

The parameter $\langle p^2\rangle$ represents the quantum average of p^2, i.e.

$$\left\langle p^2 \right\rangle \equiv \int d^3x \;\; \psi^* p^2 \psi \tag{38}$$

where

$$p^2 \equiv -\hbar^2\nabla^2 \tag{39}$$

The above equations compared to equations (2) lead to

$$\mathcal{H} = \frac{\left\langle p^2 \right\rangle}{2m} \tag{40}$$

The second statistical quantity that should be introduced now is the quadratic deviation of the position. It is given by

$$\left\langle |x - \langle x\rangle|^2 \right\rangle = \left\langle x^2 \right\rangle - \langle x\rangle^2 \tag{41}$$

where the quantum average (31) reduces to

$$\langle x\rangle = \int d^3x \;\; \rho(x)x \tag{42}$$

which is proportional to $\mathcal{G}$ as given by equation (9), and to

$$\left\langle x^2 \right\rangle \equiv \int d^3x \;\; \rho(x)x^2 \tag{43}$$

The above reductions, obviously, come from the fact that $\psi^* x\psi \;\; = \rho(x)x$ and $\psi^* x^2\psi = \rho(x)x^2$.

The parameter $\langle |x - \langle x\rangle|^2\rangle^{1/2}$represents the dispersion of the distribution ρ(x), i.e. the width of that function.

Here, however, I shall depart from the usual approach and consider another statistical parameter which also measures the width of the distribution ρ (x). This is the Fisher length [12] associated to the Fisher information associated to ρ (x) . The later is given by

$$F \equiv \int d^3x \;\; \rho(x)|\nabla \ln \rho(x)|^2 \tag{44}$$

A simple calculation leads to

$$F = 4\int d^3x \;\; \left|\nabla \rho(x)^{1/2}\right|^2 \tag{45}$$

The Fisher length, Δx, is defined by

$$\Delta x \equiv \frac{1}{F^{1/2}} \tag{46}$$

It is not a true quantum average as defined in equation (35). However, this length denotes the distance on which the distribution $\rho(x)$ has a significant variation and, as such, describes also the incertitude on the localization of the particle. Actually, as we now show, this quantity is more natural for characterizing the precision of position measurement than $\langle |x - \langle x\rangle|^2\rangle$. This is due to the property that $\mathcal{K}$ can be expressed in terms of $\langle p^2\rangle$and Δx, whereas $\mathcal{K}$ is not expressible in term of $\langle |x - \langle x\rangle|^2\rangle$. Indeed, K can obviously be written as

$$\mathcal{K} = \frac{\left\langle p^2\right\rangle}{2m} - \frac{\hbar^2}{m}\int d^3x \;\; \left|\nabla \rho(x)^{1/2}\right|^2 \quad \text{or} \quad \mathcal{K} = \frac{\left\langle p^2\right\rangle}{2m} - \frac{\hbar^2}{4m\Delta x^2} \tag{47}$$

Both relations (40) and (47) provide $\langle p^2\rangle$ and Δx^2 in terms of $\mathcal{H}$ and $\mathcal{K}$. With these relations at hand, we easily derive that the transformations (27) and (28) induce changes in $\langle p^2\rangle$and Δx^2. They represent modifications of the observer's precision:

$$\Delta x^2(\alpha) = \Delta x^2[\frac{1 - \tanh\alpha}{1 + \tanh\alpha}]^{1/2} \tag{48}$$

$$\left\langle p^2 \right\rangle (\alpha) = \frac{(1-\tanh\alpha)\left\langle p^2 \right\rangle + \frac{\hbar^2}{2}\tanh\alpha \frac{1}{\Delta x^2}}{\left[1-\tanh^2\alpha\right]^{1/2}} \tag{49}$$

The above transformation relates the precision of an observer with α equal to zero to the precision of an observer with α different from zero. The transformations of Δx and of $\langle p^2 \rangle^{1/2}$ mix both quantities. In the classical limit where $\hbar$ tends to zero, the transformations (48), (49) become simple dilatations affecting separatly Δx and $\langle p^2 \rangle^{1/2}$. Hence, in this limit there is no constraint in reducing the precision of position and momentum measurement. In the quantum case, that is when $\hbar$ is different from zero, this is no longer possible: The Heisenberg principle introduces an obstruction to this reduction. This fact is enlighten by calculating the product of the two quadratic deviations, $\Delta x^2 \langle p^2 \rangle$. One is led to the following revealing law of transformations:

$$\Delta x^2(\alpha)\left\langle p^2 \right\rangle (\alpha) = \frac{(1-\tanh\alpha)\,\Delta x^2 \left\langle p^2 \right\rangle + \tanh\alpha \frac{\hbar^2}{2}}{1+\tanh\alpha} \tag{50}$$

The quantity $\Delta x^2 \langle p^2 \rangle$ is similar to the velocity in the Lorentz transformations. The lowest value attained by this quantity is $\frac{\hbar^2}{4}$, when $\tanh\alpha = 1$. One recognizes, here, the Heisenberg inequality: $\Delta x \langle p^2 \rangle^{1/2} \geqslant \frac{\hbar}{2}$. Furthermore, for any value of α, $\Delta x^2(\alpha)\langle p^2 \rangle(\alpha)$ is equal to $\frac{\hbar^2}{4}$ if $\Delta x^2 \langle p^2 \rangle$ is already equal to $\frac{\hbar^2}{4}$. Hence, the transformation saturates at the lower limit of the Heisenberg inequality.

We, thus, have unveiled a structure in quantum mechanics that reflects a relativity of precision. Observers appear as endowed of a characteristic that is the product precision, $\Delta x^2 \langle p^2 \rangle$,whose minimum value is bounded and the same for all the observers, $\frac{\hbar^2}{4}$.It clearly appears from (48), (49) and (50) that in quantum mechanics the constant $\frac{\hbar}{2}$ plays a role similar to that of the velocity of light in special relativity. The relativity of precision so discovered has some analogy with the scale relativity appearing in the work of L.Nottale [3] which is a consequence of the basic postulate made by this author about the fractality of space-time. In our approach, however, this relativity structure is not postulated. It is intrinsically related to the algebraic structure corresponding to the twin unitary and non-unitary groups that has been put in light in this work. Some physical flesh is given to the time variable τ by this structure. Indeed, the existence of the above precision transformations and, consequently, the universality of the Heisenberg inequalities for all the observers, relies on the existence of this new time variable.

5 Conclusion

Quantum mechanics appears through our results to possess a structure of relativity theory based on the precision of measurements of the observers. This corresponds to an algebraic structure involving two twin algebras, one generating the usual unitary group, while the other generates a non-unitary group. Geometrically, the consequence of this double algebra is the existence of a second time dimension. The physical nature of this second temporal

dimension is confirmed by its link with the precision transformations that are induced by Lorentz-like transformations in both time dimensions. Evolution in this time variable is non-unitary.
More precisely, we will show in a subsequent article [10] that the evolution equation for the wave function in time τ is a nonlinear Schrödinger equation generated by the functional $\mathcal{K}$. This aspect is not discussed in the present article for the sake of conciseness and is still under study. However, recent calculations indicate that the solutions of this nonlinear Schrödinger equation exhibit a collapse of the wave function in many situations. For example, the width of an arbitrary Gaussian packet at time τ=0 always decreases in time τ.
Such time evolution presents some similarity with the measurement process which is instantaneous in time t and does not correspond to a unitary process. I suggest the hypothesis that the collapse is not instantaneous in time τ, while it is in time t. If this is the case, the collapse would be a non-unitary process governed by the above refered nonlinear Schrödinger equation. The non-unitary aspect of the wave function collapse is also underlined by Penrose [9] who advocates that some nonlinear evolution equation should describe this process. The difference, however, between his approach and the present one is double. First, in his approach the nonlinear equation describes an evolution in time t. Second, the nonlinearity is related to the gravitational interaction in the Penrose theory, while our results concern a non-relativistic free particle without any external field.
The existence of a second time dimension and the status of quantum mechanics as a relativity of precision is not without consequence with respect to the unification of gravity with quantum mechanics. Indeed, these new aspects of the quantum physics are shrinking the gap between both theories. Both rely on the notion of observer and on the transformations that connect them while keeping invariant the laws of physics. Moreover, both theories are geometrically based on the physical space-time even though, as we have shown, quantum mechanics implies a supplementary time dimension.
We are currently studying the relativistic generalisation of the above results. Preliminary work seem to indicate that the Klein-Gordon and Dirac theories do present the same algebraic duality between a unitary and a non-unitary group of transformations. They both appear to involve a structure of precision relativity as in the non-relativistic case. However, more work must be carried out in order to determine the complete algebraic structure that encompasses and generalizes both relativity groups, the special relativistic and the quantum ones.

6 Acknowledgement

It is a pleasure to be invited to the TC 2006 Conference in honor of Jean DellaDora. I would like to take advantage of this occasion to wish him a happy future and many more scientific adventures.
I am also indebted to Drs. R.Balescu, I.Veretennicoff, C.Georges, J.Reignier, C.Schomblondt, G.Barnich, R.Lambiotte and Mr.F.Ngo for the fruitful scientific discussions they had with me during this work.

References

[1] E. Santamato, Geometric derivation of the Schrödinger equation from classical mechanics in curved Weyl spaces, Physical Review D **29** (1984), 216–222.

[2] A. Shojai and F. Shojai, Constraints Algebra and Equations of Motion in the Bohmian Interpretation of Quantum Gravity, Classical and Quantum Gravity **21** (2004), 1.

[3] L. Nottale, *Fractal Space-Time and Microphysics*, World Scientific , Singapore, 1993.

[4] D. Bohm and B. J. Hiley, Measurement Understood Through The Quantum Potential Approach, Foundations of Physics **14** (1984), 255.

[5] E. Nelson, *Quantum Fluctuations*, Princeton University Press, New Jersey, 1985.

[6] F. Guerra and R. Marra, Origin of the quantum observable operator algebra in the frame of stochastic mechanics, Physical Review D **28** (1983), 1916–1921.

[7] R. Carroll, *Fluctuation, Information, Gravity and the Quantum Potential*, Springer, Berlin, 2005.

[8] R. Carroll, Fluctuations, Gravity and the Quantum Potential, arXiv **gr-qc/0501045 v1** (2005).

[9] R. Penrose, *The Road to Reality*, Jonathan Cape, London, 2004.

[10] L. Brenig, Quantum mechanics: A relativity of measurement precision, in preparation (2006).

[11] E. Sudarshan and N. Mukunda, *Classical Dynamics: A Modern Perspective*, Robert E.Krieger Pub.Company, Malabar, 1983.

[12] M. J. Hall, Quantum properties of classical Fisher information, Physical Review A **62** (2000), 012107-1.

Léon Brenig
Faculté des Sciences. Université Libre de Bruxelles.
Campus de la Plaine, CP231, boulevard du Triomphe, 1050 Bruxelles, Belgique.
lbrenig@ulb.ac.be

André-Louis Cholesky: his life and works

Claude Brezinski

Introduction

Let A be a symmetric positive definite matrix. It can be decomposed as $A = LL^T$ where L is a lower triangular matrix with positive diagonal elements. Then, the system $Ax = b$ writes $LL^Tx = b$. Setting $y = L^Tx$, we have $Ly = b$. Solving this lower triangular system gives the vector y. Then x is obtained as the solution of the upper triangular system $L^Tx = y$. This is Cholesky's method.

Cholesky died in 1918. His famous method for systems of linear equations was only known from a paper published in 1924 by one of his fellow officer, the Commandant Benoît.

In 1995, the archives of the army were opened to the public, and I wrote his first complete biography [2].

Last year, I was contacted by Cholesky's grand-son who gave all papers he had to École Polytechnique were his grand-father studied, and he asked me to help him to classified them. Then, we found the original manuscript where Cholesky described his method.

1 André-Louis Cholesky

André Louis Cholesky was born on October 15, 1875 in Montguyon, around 35km nord-east of Bordeaux. He was the son of André Cholesky, head waiter, and of Marie Garnier. He went to the high school in Saint-Jean-D'Angély. He obtained his "baccalauréat" in Bordeaux in 1893.

In October 1895, he entered as a student at École Polytechnique for 2 years. His professors were Camille Jordan and Georges Humbert for analysis, Émile Haag for geometry, Octave Callandreau for astronomy and geodesy, and Henri Becquerel for physics. In October 1897, after finishing École Polytechnique, he became Sous-Lieutenant, and studied at the École d'Application de l'Artillerie et du Génie in Fontainebleau. Among others, he had lectures on ballistic and topography. He ended 5th over 86.

On October 1st, 1899, he became Lieutenant in the 22th Régiment d'Artillerie. From January 17 to June 27, 1902 he was sent to Tunisia and again from November 21, 1902 to May 1st, 1903. From December 1903 to June 1906, he was in Algeria. On June 24, 1905, he was appointed to the Geographical Service of the Headquarters of the Army. He was in the section having to mesure the length of the meridian of Lyon. On May 10, 1907 he married his first cousin Anne Henriette Brunet, born June 27, 1882. They will have 4 children.

Cholesky went to Crete, then occupied by the international troops, from November 7, 1907 to June 25, 1908. He had to cartography the British and French parts of the island. On March 25, 1909, he was promoted to Captain. On August 28, 1909, he had to serve as the head of a battery for two years. It is during this period that he found his method for the solution of systems of linear equations. In October 1911, he was again appointed to the Geographical Service of the army as head of surveying in Tunisia and Algeria. He stayed in these countries until August 2, 1914, the day of the mobilization. He was head of the Topographical Service in Tunis.

After 3 months as the head of a battery, he was appointed at the Geographical Service and worked on canvas for shooting. From September 1916 to February 1918, he was sent to Romania and became head of the Geographical Service that he completely reorganized. On July 7, 1917 he was promoted Commandant. In June 1918, he was appointed at the 202th Regiment of Artillerie of the Army of General Mangin. This army was fighting in Picardie between the river Aisne and Saint-Gobain.

On August 31, 1918 Cholesky was killed north of Bagneux, a small village around 10km north of Soissons. He was first buried in the cemetery Chevillecourt near Autrèches, 15km east of Soissons. On October 24, 1921 his grave was transferred to the cemetery of Cuts, 10km south-east of Noyon.

2 Works

Cholesky was a topograph in the Geographic Service of the Army. His work mainly consisted in drawing maps of various scales. From December 1909 and, at least, until January 1914 he was also a Professor at the École Spéciale des Travaux Publics, du Bâtiment et de l'Industrie founded in 1891 by Léon Eyrolles. The studies were only by correspondence and Cholesky had to write lecture notes for the students.

In the archives, we found many manuscripts which, in fact, were incorporated into a book he published around these years [5]. It contains 442 pages, 100 figures and 18 photos of instruments. This book was quite successful since it had, at least, 7 editions and was still in the catalogue of the publisher 30 years after Cholesky's death.

The archives also contain another book with the title *Cours de Calcul Graphique*, several manuscripts describing the use of instruments for topography, and many military document on the organization of the work of a geographical officer, on shooting, etc.

3 The manuscript

Let us describe the contents of the manuscript of Cholesky entitled *Sur la résolution numérique des systèmes d'équations linéaires*, and dated December 2, 1910.

Starting from a system of linear equations, Cholesky began to multiply by the transpose of the matrix in order to obtain a system with a symmetric matrix (the matrix is also positive definite, a property which Cholesky does not mention). Then, he introduces intermediate unknowns satisfying a system with a lower triangular matrix. Then, these unknowns are

used as the right hand side of another system with a upper triangular matrix, which is the transpose of the lower triangular matrix, and whose solution is the one we are looking for. Then, by multiplying these two triangular matrices together, and identifying with the corresponding elements of the original matrix, he obtained the expressions of their elements exactly as we all taught them to our students.

Then, Cholesky discusses the implementation of his method on the mechanical machine *Dactyle*, in use at that time. He said that the full capabilities of the machine have to be used, and that sign errors could easily be recognized.

After that, he claims that, in his method, the two triangular matrices are not necessarily transpose, but he proves that this choice limits the propagation of rounding errors. Then, he gave a method for computing square roots, which is, in fact, the method of Hero of Alexandria, and he proved its quadratic convergence. A method for checking the solution of the system is then given. Finally, he reports numerical examples: 4 or 5 hours were necessary for solving a system of dimension 10 with 5 exact decimal digits. He said that his method was also used for a system of 56 equations coming out from the triangulation of Algeria.

4 Historical context

The least squares method was first published by Adrien Marie Legendre (Paris, 1752 - Paris, 1833) in 1805. Its justification as a statistican procedure is due to Carl Friedrich Gauss (Braunschweig, 1777 - Göttingen, 1855) en 1809. According to him, the method lead to the best possible combination of observations regardless of the probability law of errors. It was immediately recognized as a major contribution. Gauss maintained that he already used it in 1795. It is sure that he used it in 1801 to determine the orbit of the comet Cérès.

The American mathematician of Irish origin Robert Adrain (Carrickfergus, Ireland, 1775 - New Brunswick, USA, 1843) exposed the least squares method in a paper on topography dated 1808. This work seemed to have attracted no attention in Europ. In 1818, Adrain again used this method to determine the shape of the Earth from the measurements of the meridian and obtained estimated values for the axes of the terrestrial ellipse.

A system of linear equations has infinitely many solutions when the number of unknowns is greater than the number of equations. Among all possible solutions, one look for the solution minimizing the sum of the squares of the unknowns. This is the case in the compensation of the triangles in topography in which Cholesky was interested.

Methods for the solution of systems of linear equations were rediscovered many times by different people. They are, in fact, variations on the way to present the computation. Gaussian elimination corresponds to a decomposition of the matrix A of the system into a product $A = LU$ where L is a lower triangular matrix with a unit diagonal, and U an upper triangular matrix. The system $Ax = b$ then writes $LUx = b$, that is $Ly = b$ if we set $y = Ux$. Solving $Ly = b$ gives the vector y which is then used as the right hand side of the system $Ux = y$, and the solution x is obtained.

On November 9, 1878, Myrick H. Doolittle (Addison, USA, 1830 - 1913), a mathemati-

cian in the Computing Division of the U.S. Coast and Geodetic Survey in Washington, gave a method for the solution of systems of equations coming out from a triangulation problem. Its method consisted to cancel out, one by one, the elements of the matrix to transform it into an upper triangular one. His method is equivalent to the decomposition of A into a product $A = LU$ by a sequence of intermediate steps. He obtained these matrices under the form $L = L_1 + \cdots + L_n$ and $U = U_1 + \cdots + U_n$. L is lower triangular and U is upper triangular, but, contrarily to Gauss method, with a unit diagonal. Doolittle had no machine at his disposal, and he simply used multiplication tables. He said to have solved a system of 41 equations in 5 and a half days, corresponding to 36 hours of computation. His method had the favor of geodesists for many years.

When the matrix A is symmetric, Gauss method makes no use of this property, and requires too many arithmetical operations. In 1907, Otto Toeplitz (Breslau, Allemagne, 1881 - Jerusalem, 1940) showed that an Hermitian matrix can be factorized into a product LL^* with L lower triangular, but he gave no algorithm for obtaining the matrix L. This is what Cholesky did in 1910. Cholesky method remained unknown outside the circle of French military topographers.

Other scientists continued to propose variants of the methods of Gauss and Doolittle in order to make the computation easier for people without a mathematical culture. In 1938, the Polish astronomer Tadeusz Banachiewicz (Varsaw, 1882 - 1954) proposed a square root method very similar to Cholesky's. He used cracovians, objects he had invented and are similar to matrices but with a different product rule. Banachiewicz was the first to stress that elimination methods for the solution of systems of linear equations were equivalent to the factorization of the matrix A into a product of two matrices. But, as we saw, he had been preceded by Cholesky.

In 1941, Paul Summer Dwyer (born in 1901) gave an abbreviate version of Doolittle method and related it to other solution methods. In 1944, he gave the matrix interpretation of Doolittle method. He also showed that $L = DU^T$ where D is a diagonal matrix and noticed that it will be more interesting if L and U^T were the same, in order to reduce the number of arithmetical operations. For that, one can take the square roots of the diagonal elements, that is the elements of D, and he noticed that this procedure was quite similar to Banachiewicz's.

Cholesky method was presented for the first time in 1924 in a note by Commandant Benoît, a French geodesist.

Cholesky method was rebirth by John Todd who taught it in his numerical analysis course at King's College in London in 1946 and thus made it known.

He tells [6]

> *In 1946 one of us (J.T.) offered a course at Kings' College, London (KCL) on Numerical Mathematics. While we had some wartime experience in numerical mathematics, including characteristic values of matrices, we had had little to do with the solution of systems of linear equations. In order to see how this topic should be presented, we made a survey of Math. Rev. (at that time easy!) and found a review (MR7 (1944), 488), of a paper by Henry Jensen, written by E.*

Bodewig. Jensen stated "Cholesky's method seems to possess all advantages." So, it was decided to follow Cholesky and, since the method was clearly explained, we did not try to find the original paper.

Leslie Fox (1918–1992), then in the newly formed Mathematics Division of the (British) National Physical Laboratory (NPL), audited the course and apparently found the Cholesky Method attractive, for he took it back to NPL, where he and his colleagues studied it deeply. From these papers, the Cholesky (or sometimes Choleski) Method made its way into the tool boxes of numerical linear algebraists via the textbooks of the 1950's.

His colleagues were James H. Wilkinson and Alan M. Turing.

For more details on the life and works of Cholesky and his full manuscript, see [3, 4].

Bibliography

[1] Cdt. Benoît, Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues, (Procédé du Commandant Cholesky), Bulletin Géodésique, 2 (1924) 67-77.

[2] C. Brezinski, André Louis Cholesky, dans *Numerical Analysis, A Numerical Analysis Conference in Honour of Jean Meinguet*, Bull. Soc. Math. Belg., 1996, pp. 45-50.

[3] C. Brezinski, La méthode de Cholesky, Rev. Hist. Math., à paraître.

[4] C. Brezinski, M. Gross-Cholesky, La vie et les travaux d'André Louis Cholesky, Bull. Soc. Amis Bibl. Éc. Polytech., 39 (2005) 7-32.

[5] A. Cholesky, *Cours de topographie. 2è partie : Topographie générale*, 7è éd., École Spéciale des Travaux Publics, Paris, 1937.

[6] O. Taussky-Todd, J. Todd, Cholesky, Toeplitz and the triangular factorization of symmetric matrices, Numer. Algorithms, to appear.

Claude Brezinski
Laboratoire Paul Painlevé, UMR CNRS 8524
UFR de Mathématiques Pures et Appliquées
Université des Sciences et Technologies de Lille
59655 - Villeneuve d'Ascq cedex
France
claude.brezinski@univ-lille1.fr

From genomic signatures to genomic functional cores

Alessandra Carbone

Abstract

The project of synthesizing a bacterial genome which can survive in the laboratory and can realize desired metabolic cycles, has been announced three years ago by Venter, Smith and Hutchison. It asks for clearing out certain basic biological mechanisms of living cells. The problem demands to search for the minimal set of genes that are essential to the life of a microbial organism. Laboratory experiments realized on specific bacteria allowed to propose some minimal gene set. Independently, comparative genomics also proposed some minimal set of genes. But both these "solutions" present some intrinsic problem.

We shall present some simple mathematical ideas based on Gibbs sampling, that allow to detect genomic signatures for sets of genes and sets of organisms, and to predict genomic functional cores which are specific to different microbes. Within these sets, one finds many of the genes characterized with experiments and genome comparison, but also genes which might be non-orthologous or whose function might not be characterized yet. More generally, our computational approach leads to characterize essential metabolic pathways through a purely statistical analysis of complete genomes which is independent from biological assumptions.

Alessandra Carbone
Génomique Analytique, Université Pierre et Marie Curie
INSERM U511, 91, bd de l'Hôpital. 75013 Paris France
Alessandra.Carbone@lip6.fr

Formal and numerical computation of invariants: from differential equations to q-difference equations

Jean-Pierre Ramis

Abstract

We will review invariant theory for linear differential equations (algebraic and analytical invariants, local and global invariants) and compare with the recent improvements for the parallel case of linear q-difference equations.

In the differential case there exists a lot of effective computations (formal and numerical) initiated in an old collaboration with Jean Della Dora and his students (DESIR...). In the q-difference case there are some similar algorithms (in ORE style) but also a lot of open problems.

Jean-Pierre Ramis
Laboratoire Émile Picard. Université Paul Sabatier
118 route de Narbonne. 31062 Toulouse Cedex 4
ramis@picard.ups-tlse.fr

Algebraic Machines

Tomás Recio Muñiz

Abstract

This talk deals with the prehistory (from the point of view of the speaker) of Computer Algebra. It will provide some evidences (already pointed out by different authors, such as P.J. Larcombe, "On Lovelace, Babbage and the origins of computer algebra", in Computer Algebra Systems, A practical guide. Edited by Michael Wester, J. Wiley. 1999. pp. 323-331) on the connection between Computer Algebra and the well known work of Babbage.

But it will also refer to the less known contribution of a spanish scientist, in the last years of the XIX-th century, who published some of his results at the CR Acad. Sc. Paris (perhaps the first spanish accepted submission to this journal), refereed by Poincaré himself... In the talk we will briefly mention some aspects of his life and activities (some of them probably well known by many americans living near Niagara Falls, but yet ignorant of their connection to Computer Algebra).

One of his most cherished goals (one that he also shares with Babbage) connects the origins of Computer Algebra with the early days of industrial revolution (back in the XVIII-th century). We will present some historical digressions around this topic (mentioning, among other issues, the British Parliament and a bitter Russian-French military dispute on the authoring of one of the main results) and we will display some interactive examples, by visiting some interesting internet sites.

Finally we will pay attention to the traces, in today's mathematics and computing, of all this history, referring to a universality theorem attributed to Thurston (but proved by someone else). Algebraic machines are indeed universal!

Tomás Recio Muñiz.
Departamento de Matemáticas, Estadística y Computación.
Facultad de Ciencias, Universidad de Cantabria.
Avenida de los Castros, s/n 39071 Santander, España.
Tomas.Recio@unican.es

Making Computer Algebra More Symbolic

Stephen M. Watt

Abstract

This paper is a step to bring closer together two views of computing with mathematical objects: the view of "symbolic computation" and the view of "computer algebra." Symbolic computation may be seen as working with expression trees representing mathematical formulae and applying various rules to transform them. Computer algebra may be seen as developing constructive algorithms to compute algebraic quantities in various arithmetic domains, possibly involving indeterminates. Symbolic computation allows a wider range of expression, while computer algebra admits greater algorithmic precision. We examine the problem of providing polynomials symbolic exponents. We present a natural algebraic structure in which such polynomials may be defined and a notion of factorization under which these polynomials form a UFD.

1 Introduction

For the purposes of this paper, it is useful to make a distinction between "symbolic computation" and "computer algebra."

By "symbolic computation," we mean computation with expression trees, or "terms," representing mathematical objects. In these trees may appear symbols denoting operations, such as "+," "×" or "sin", numbers and variables. Computation consists of combining or transforming these trees to, for example, expand multiplications or multiple angle formulae. There are problems related to expression equivalence, simplification and computing canonical forms. A given expression might represent values belonging to various mathematical domains, depending on the interpretation. For example, is "I" an identity matrix, an indexed Bessel function, an imaginary unit, etc. Algebraic algorithms are typically well-defined on subclasses of the space of all expressions. Working with new classes of objects requires defining new operators and transformations on expressions containing them.

By "computer algebra," we mean computations using the arithmetic from particular algebraic constructions. The values used are elements of mathematically defined sets, such as polynomial rings, algebraic extensions, quotients and so on. The elements might be represented in any one of a number of ways. Certain algebraic domains, such as polynomials, may include indeterminates. Algorithms are defined over particular algebraic input domains and yield well-defined results. Working with new classes of objects requires defining new algebraic domains and determining their properties.

A useful problem solving environment should provide some way to do both symbolic computation and computer algebra, and there are different ways to do this. We may view, for example, Maple as being a symbolic computation system with computer algebra built as a layer on top. On the other hand, we may view Axiom as a computer algebra system, with symbolic computation provided as a top layer. In both cases we suffer because there is a gap between the symbolic and the algebraic semantics.

This gap is particularly evidenced when we work problems by hand. We do not hesitate to work with vectors of dimension n, or write polynomials of degree d with coefficients from some ring k of characteristic p. We then take facts about d, n, k or p into account when we do calculations. As important as it is, there is relatively weak support for this sort of computation in our symbolic mathematical software.

We are motivated to explore how to bring the symbolic and algebraic views closer together, providing a more robust conceptual framework and providing tools to address an important family of practical problems. We may view this as defining algebraic domains for wider classes of symbolic expressions. That is, we wish to work in algebraic domains that lie beyond the well-studied algebraic constructions of classical algebra. As a start, we examine the problem of working with polynomials with symbolic exponents.

2 Symbolic Polynomials

We wish to work with polynomials where the exponents are not known in advance, such as $x^{2n}-1$. There are various operations we will want to be able to do, such as squaring the value to get $x^{4n}-2x^{2n}+1$, or differentiating it to get $2nx^{2n-1}$. This is far from a purely academic problem. Expressions of this sort arise frequently in practice, for example in the analysis of algorithms, and it is very difficult to work with them effectively in current computer algebra systems.

We may think of these values as sets of polynomials, one for each value of n, or we may think of them as single values belonging to some new ring. We wish to perform as many of the usual polynomial operations on these objects as possible. Many computer algebra systems will allow one to work with polynomials with symbolic exponents. They do this, however, either by falling back on some form of general expression manipulation or by treating all symbolic powers as algebraically independent. They thus miss many of the important properties we wish to reflect. The relationship between exponents may be non-trivial. We would like, for example, to compute factorizations such as

$$x^{n^4-6n^3+11n^2-6(n+2m-3)}-1000000^m$$

$$\begin{aligned} &= x^{-12m}\times\left(x^{p_1}+10^m x^{p_2+2m}+10^{2m}x^{4m}\right)\times\left(x^{p_2}+10^m x^{2m}\right)\\ &\qquad\times\left(x^{p_1}-10^m x^{p_2+2m}+10^{2m}x^{4m}\right)\times\left(x^{p_2}-10^m x^{2m}\right)\end{aligned}$$

$$\begin{aligned} p_1 &= x^{1/3n^4-2n^3+11/3n^2-2n+6}\\ p_2 &= x^{1/6n^4-n^3+11/6n^2-n+3}\end{aligned}$$

and perhaps operations on symbolic integers

$$16^n - 81^m = (2^n - 3^m)(2^n + 3^m)(2^{2n} + 3^{2m}).$$

We can imagine a number of models for symbolic polynomials that have these properties. Most generally, we could say that any set S, which under an evaluation map ϕ gives a polynomial ring $R[x_1, ..., x_v]$, represents symbolic polynomials. This would allow such forms as

$$\gcd(x^n - 1, x^m - 1) - x^{\mathrm{lcm}(n,m)} + 1$$

or

$$(x - 1)\sum_{i=0}^{n} x^i.$$

Having a more obvious ring structure will be useful to us, so we begin by generalizing to symbolic exponents only. First we recall the concept of a "group ring." A *monoid ring* is a ring formed from a ring R and monoid M with elements being the finite formal sums

$$\sum_i r_i m_i, r_i \in R, m_i \in M.$$

A monoid ring has a natural module structure, with basis M, and addition defined in terms of coefficient addition in R. Multiplication is defined to satisfy distributivity, with $r_1 m_1 \times r_2 m_2 = (r_1 r_2)(m_1 m_2)$. When the monoid M is a group, then the algebraic structure is called a *group ring*. For example, the Laurent polynomials with complex coefficients may be constructed as the group ring $\mathbb{C}[\mathbb{Z}]$, viewing $\mathbb{Z}$ as an additive group.

We now define a useful class of symbolic polynomials.

Definition 2.1. *The ring of* symbolic polynomials *in $x_1, ..., x_v$ with exponents in $n_1, ..., n_p$ over the coefficient ring R is the ring consisting of finite sums of the form*

$$\sum_i c_i x_1^{e_{i1}} x_2^{e_{i2}} \cdots x_n^{e_{in}}$$

where $c_i \in R$ and $e_{ij} \in \mathrm{Int}(\mathbb{Z})[n_1, n_2, ..., n_p]$. Multiplication is defined by

$$c_1 x_1^{e_{11}} \cdots x_n^{e_{1n}} \quad \times \quad c_2 x_1^{e_{21}} \cdots x_n^{e_{2n}} = c_1 c_2 x_1^{e_{11}+e_{21}} \cdots x_n^{e_{1n}+e_{2n}}$$

We denote this ring $R[n_1, ..., n_p; x_1, ..., x_v]$.

We make use of the notion of "integer-valued polynomials," $\mathrm{Int}(D)[n_1, ...n_p]$. For an integral domain D with quotient field K, univariate integer-valued polynomials may be defined as

$$\mathrm{Int}(D)[X] = \{f(X) \,|\, f(X) \in K[X] \text{ and } f(a) \in D, \text{ for all } a \in D\}$$

For example $\frac{1}{2}n^2 - \frac{1}{2}n \in \mathrm{Int}(\mathbb{Z})[n]$. Integer-valued polynomials have been studied by Ostrowski [2] and Pólya [3], and we take the obvious multivariate generalization.

Our definition of symbolic polynomials is isomorphic to the group ring $R[\mathbb{Z}[n_1, ..., n_p]^v]$. We view $\mathbb{Z}[n_1, ...n_p]$ as an abelian group under addition and use the identification

$$X_1{}^{e_1} X_2{}^{e_2} \cdots X_v{}^{e_v} \cong (e_1, \ldots, e_v) \in \mathbb{Z}[n_1, \ldots, n_w]^v$$

We note that $R[; x_1, ..., x_v] \cong R[x_1, ..., x_v]$. Under any evaluation $\phi : \{n_1, ..., n_p\} \to \mathbb{Z}$, we have

$$\phi : R[n_1, ..., n_p; x_1, ..., x_v] \to R[x_1, ..., x_v, x_1^{-1}, ..., x_v^{-1}].$$

That is, ϕ evaluates symbolic polynomials to Laurent polynomials. It would be possible to construct a model for symbolic polynomials that, under evaluation had no negative variable exponents, but this would require keeping track of cumbersome domain restrictions on the exponent variables.

By definition, these symbolic polynomials have a ring structure. What is more interesting is that they also have a useful unique factorization structure that can be computed effectively.

3 Factorization

We now show the multiplicative structure of our symbolic polynomials, and describe two algorithms for factorization. For simplicity we first show the case when $R = \mathbb{Z}$ and exponents are in $\mathbb{Z}[n_1, ..., n_p]$.

Proposition 3.1. *$\mathbb{Q}[n_1, ..., n_p; x_1, ..., x_v]$ is a UFD, with monomials being units.*

We sketch the proof: The fact that $x, x^n, x^{n^2}, ...$ are algebraically independent can be used to remove exponent variables inductively. We observe that

$$x_k^{e_{ik}} = x_k^{\sum_j h_{ij} n_1^j} = \prod_j \left(x_k^{n_1{}^j}\right)^{h_{ij}} = \prod_j x_{kj}{}^{h_{ij}}, \qquad h_{ij} \in \mathbb{Z}[n_2, ..., n_p].$$

This gives the isomorphism

$$\mathbb{Z}[n_1, n_2, ..., n_p; x_1, ...x_v] \cong \mathbb{Z}[n_2, ..., n_p; x_{10}, x_{11}, x_{12}, ...x_{1d_1}, ...x_{v0}, x_{v1}, x_{v2}, ...x_{vd_1}]$$

where d_1 is the maximum degree of n_1 in any exponent polynomial and x_{ij} corresponds to $x_i^{n_1{}^j}$. Once all the exponent variables have been removed, we factor in $\mathbb{Z}[x_{10...0}, ..., x_{vd_1...d_1}]$, then reform the exponent polynomials of $x_1, ..., x_v$.

When the exponents come from the integer-valued polynomials $\mathrm{Int}(\mathbb{Z})[n_1, ..., n_p]$, as opposed to $\mathbb{Z}[n_1, ...n_p]$, care must be taken to find the fixed divisors of the exponent polynomials. For example, the fact that $n(n-1)$ is always even implies the factorization

$$x^2 - y^{n^2-n} = (x - y^{n(n-1)/2})(x + y^{n(n-1)/2})$$

Fixed divisors are given by the content when polynomials are written in a factorial basis. That is, for each exponent variable n_i, using the polynomial basis $1, n_i, n_i^{\underline{2}}, ..., n_i^{\underline{j}}, ...$, where

$$n^{\underline{j}} = \frac{n(n-1)\cdots(n-j+1)}{j!}.$$

Combining these two ideas, we make the change of variables to $x_i^{n_j^{\underline{k}}}$, to obtain factorization in $\mathbb{Q}[n_1, ...n_p; x_1, ..., x_v]$. The same strategy may, of course, be used to compute greatest common divisors, square-free factorizations and similar quantities.

We have described this transformation as though the exponent polynomials were dense. In this worst case, the number of new variables will be D^p, where D is the degree bound on the n_i. In practice, the number of variables occurring in exponents will be small and the exponent polynomials will be of low degree so the introduction of new variables may be acceptable. In many cases, most of the new variables will occur only trivially. Blindly changing to a factorial basis to make fixed divisors manifest may not, however, be the best strategy. This destroys any sparseness in the input polynomials. A better strategy would be to convert only when necessary.

If the number of exponent variables is large, then another method may be used to manage the complexity of many variables. We may use projections to map the exponents to integers at several points, and combine them via interpolation. Naively, an exponential number of factorizations in $\mathbb{Q}[x_1, ..., x_v]$ will be needed, but this not always necessary. If there are may small factors, then there is a combinatoric problem of factor identification. If the coefficient field is large enough, and the polynomials are not special, then coefficient values may be used to greatly limit the search. We have experimental implementations of both the "change of variables" method and the "projection" method, but it is too early to say which method will be most useful in practice.

4 Generalizations

As mentioned earlier, we may contemplate other algebraic structures to encompass a wider class of expressions. Without going to the most general model of polynomial-valued integer functions, we may consider

- Allowing exponent variables to also appear as regular variables. To do this we can work in $R[n_1, ..., n_p; n_1, ..., n_p, x_1, ..., x_v]$. This is useful if we require formal derivatives.
- Symbolic exponents on coefficients. We discuss this case more below.
- Symbolic polynomials as exponents, or richer structures.
- Other polynomial forms, such as exponential polynomials, *e.g.* [1] [4].
- Other problems, *e.g.* Gröbner bases of symbolic polynomials [5].

Let us examine more closely the question of symbolic exponents on coefficients. Suppose we wish to factor a polynomial of the form $x^{4m} - 2^{4n}$. Assuming m and n may take on only integer values, the factorization over $\mathbb{Q}$ is $(x^{2m} + 2^{2n})(x^m + 2^n)(x^m - 2^n)$. This, however is equivalent to $x^{4m} - 16^n$, which is not manifestly the difference of fourth powers. So how can we approach symbolic integer coefficients?

If the coefficient ring is a principal ideal domain, then we may extend our definition to allow symbolic exponents on prime coefficient factors:

Definition 4.1. *The ring of* symbolic polynomials *with exponents in $n_1, n_2, ..., n_p$ over the coefficient ring R, a PID with quotient field K, is the ring consisting of finite sums of the form*

$$\sum_i k_i \cdot \prod_j c_j^{d_{ij}} \cdot x_1^{e_{i1}} x_2^{e_{i2}} \cdots x_n^{e_{in}}$$

where each product has a finite number of nonzero d_{ij}, $k_i \in K$, c_j are primes $\in R$, $d_{ij} \in \mathrm{Int}(\mathbb{Z})[n_1, n_2, ..., n_p] \backslash \mathbb{Z}$ and $e_{ij} \in \mathrm{Int}(\mathbb{Z})[n_1, n_2, ..., n_p]$. Multiplication is defined by

$$\begin{aligned} k_1 c_1^{d_{11}} \cdots c_m^{d_{1m}} x_1^{e_{11}} \cdots x_n^{e_{1n}} \quad \times \quad & k_2 c_1^{d_{21}} \cdots c_m^{d_{2m}} x_1^{e_{21}} \cdots x_n^{e_{2n}} = \\ & k_1 k_2 c_1^{d_{11}+d_{21}} \cdots c_m^{d_{1m}+d_{2m}} x_1^{e_{11}+e_{21}} \cdots x_n^{e_{1n}+e_{2n}} \end{aligned}$$

Let us consider the case of integer coefficients. We note that, for any base, any set of logarithms of distinct primes is linearly independent over $\mathbb{Q}$. This is easily seen, for the equation $\sum_i n_i \log(p_i) = 0$, holds with p_i distinct primes and $n_i \in \mathbb{Z}$, only if $\prod_i p_i^{n_i} = 1$, which requires $n_i = 0$. This implies that

$$\sum_i \alpha_i \log p_i \neq 0$$

for any non-zero algebraic numbers α_i. We can write any product of integers to symbolic powers as an exponential of a linear combination of logarithms of primes, *e.g.*

$$6^m \times 7^{n^2+1} = \exp(m \log 2 + m \log 3 + (n^2+1) \log 7)$$

where exp and log use the same base. We can therefore treat 2^n, 2^{n^2}, ... as new variables for factoring, etc.

As stated, this approach would require factoring each integer that appears with a symbolic exponent. In practice we do not want to factor the constant coefficients. Instead, we can form, for any particular problem, an easier to compute basis, *e.g.* from $\{70^n, 105^n\}$ the set $\{2^n, 3^n, 35^n\}$ which does not require factoring of 35. This can be done using only integer gcd's and extracting integer roots.

5 Conclusions

We see a mathematically rich and practically important middle ground between the usual approaches of "symbolic computation" and "computer algebra." Rather than working with loosely defined expressions, or strictly with classical polynomial and matrix algebras, there is room to work in other well-defined algebraic contexts. These can provide the structure to make operations well-defined, while at the same time allowing more symbolic treatment in mathematical computations. In this light, we have explored how to usefully work with symbolic polynomials — polynomial-like objects where the exponents can themselves be

polynomials. These are able to represent the kinds of symbolic polynomials we have seen in practice. The algebraic structure allows us to perform arithmetic on these objects, to simplify and transform them. We find, moreover, a UFD structure that admits algorithms for factorization, gcd, *etc.* This encourages us to look at more algebraic treatment of other symbolic structures, such as matrices of unspecified size.

References

[1] de Prony, Baron Gaspard Riche. Essai éxperimental et analytique: sur les lois de la dilatabilité de fluides élastique et sur celles de la force expansive de la vapeur de l'alkool, à différentes températures. Journal de l'École Polytechnique, volume 1, cahier 22, 24-76 (1795).

[2] Ostrowski, A., Über ganzwertige Polynome in algebraischen Zahlköpern, J. Reine Angew. Math., 149 (1919), 117-124.

[3] Pólya, G., Über ganzwertige Polynome in algebraischen Zahlköpern, J. Reine Angew. Math., 149 (1919), 97-116.

[4] C.W. Henson, L. Rubel, and M. Singer, Algebraic Properties of the Ring of General Exponential Polynomials. Complex Variables Theaory and Applications, **13**, 1989, 1-20.

[5] Kazuhiro Yokoyama. On Systems of Algebraic Equations with Parametric Exponents. pp 312-319, ISSAC '04, July 4-7, 2004, Santander, Spain, ACM Press.

Stephen M. Watt
Ontario Research Centre for Computer Algebra
University of Western Ontario, MC 375
London ON, Canada N6A 5B7

watt@orcca.on.ca
http://www.orcca.on.ca/~watt

Chapter 2

Full papers

A computational method to obtain the law of the nilpotent lie algebras $\mathfrak{g}_n$

Juan Carlos Benjumea Juan Núñez Ángel F. Tenorio

Abstract

In this paper we study the law of the Lie algebras $\mathfrak{g}_n$ associated with a particular type of Lie groups: the Lie groups G_n formed by all the $n \times n$ upper-triangular matrices with ones in the main diagonal. The Lie algebras $\mathfrak{g}_n$ are formed by all the $n \times n$ upper-triangular matrices with zeros in the main diagonal. We compute their laws by using a computational algorithm, which we have constructed and particularly implemented in MAPLE V©.

2000 Mathematics Subject Classification: Primary 17–08, 17B30; Secondary 68W30.
Key words and phrases: nilpotent Lie algebra, law of Lie algebra, algorithmic procedure, algebraic programming, computer algebra.

Introduction

At present, a very studied topic in Mathematics is the relation existing between Lie groups and Lie algebras. It is well-known that there is a unique Lie algebra associated with every Lie group given. The construction of this Lie algebra can be made by using the left-invariant differentiable vector fields (see [9], for example). On the other hand it is possible to obtain a Lie group associated with a given finite-dimensional Lie algebra (see Theorem 3.17.8 in [9]). Note, however, that it does not occur in infinite dimension, where there exist Lie algebras which are not associated with Lie groups (see [10]).

Besides, the existence of a Lie group associated with a given Lie algebra does not imply the uniqueness of such a Lie group. Indeed, such a Lie group is not unique. In page 42 of [7] it is proved that the Lie algebra associated with a given Lie group and its connected component of the identity are isomorphic each other. A more restrictive condition over the Lie group has to be imposed to assert the uniqueness of the Lie group associated with the given Lie algebra.

In this way, Lie's Third Theorem (and its converse) states that there exists a unique, up to isomorphism, simply connected Lie group associated with a given Lie algebra (see Theorem 2.8.2 in [9]). As a consequence, Lie's Third Theorem and its converse set a uniquely correspondence between simply connected Lie groups and Lie algebras. A direct proof of this result, which is known as Cartan's Theorem, uses a geometrical construction based on Maurer-Cartan constants and on the group of automorphisms. However, the accustomed

proof in the literature is based on Ado's Theorem (see [5]), which sets that given a Lie algebra $\mathfrak{g}$, there exists a linear algebra isomorphic to it.

In this paper we study the law of the Lie algebras associated with some particular Lie groups: the Lie groups G_n formed by all the $n \times n$ upper-triangular matrices with ones in the main diagonal. These Lie algebras, denoted by $\mathfrak{g}_n$, are formed by all the $n \times n$ upper-triangular matrices with zeros in the main diagonal.

The importance of the Lie algebras $\mathfrak{g}_n$ and the Lie groups G_n lies in the following fact: every nilpotent Lie algebra is isomorphic to a Lie subalgebra of some Lie algebra $\mathfrak{g}_n$ (see [4]) and, analogously, every simply connected nilpotent Lie group is isomorphic to a Lie subgroup of some Lie group G_n (see Theorem 3.6.3 in [9]).

To compute these laws, we have constructed a computational algorithm which can be implemented in any symbolic computation package, although we have particularly used MAPLE V© to do it. It constitutes the main aim of this paper. Observe that the order n of the matrices in $\mathfrak{g}_n$ is going to be the unique input data needed to obtain the law of $\mathfrak{g}_n$.

Note that the computational study of Lie algebras has been developed throughout these last four decades. Since the seventies of the 20th century, several authors have used algorithms to study the structure of Lie algebras (see [1], for instance). Moreover, in recent papers on this subject authors have even turned to specialized computational packages, like MAGMA (see [3]) or GAP (see [2]) for instance. It constitutes our motivation.

The structure of this paper is the following: the first section sums up the concepts and results used to develop and justify that algorithm, which is completely detailed in the second section. In the next section we explain all the clusters in the programming to implement the algorithm. Finally, some conclusions obtained from this implementation, like the time that it takes according to different dimensions, for instance, are shown.

1 Theoretical groundwork

We recall in this section the preliminary concepts and results on Lie algebras that will be used in the paper. For a general overview on Lie groups and Lie algebras, the reader can consult [9]. General information about commands and programming of MAPLE V© can be consulted in [6] for instance.

As the Lie algebras $\mathfrak{g}_n$ will be constructed starting from their associated Lie groups G_n, we also remind the construction of the Lie algebra associated with a given Lie group G.

The m-dimensional Lie algebra $\mathfrak{g}$ associated with a given m-dimensional Lie group is the vector space $\chi(G)$ formed by all the left-invariant vector fields endowed with the Lie-Poisson bracket defined by:

$$[X, Y] = X \circ Y - Y \circ X, \quad \forall X, Y \in \chi(G).$$

Each 1-dimensional vector space of left-invariant vector fields is associated with a one-parameter subgroup of G. Indeed, by using the one-parameter subgroups, a basis of the Lie algebra $\mathfrak{g}$ can be obtained. Recall that a *one-parameter subgroup* of G is a group homomorphism $\varphi : \mathbb{C} \to G$ defined between the additive group $\mathbb{C}$ of the complex numbers and the Lie group G.

To obtain the left-invariant vector field associated with a one-parameter subgroup φ, it is necessary to find all the orbits for such fields. These orbits are exactly the curves $g \cdot \varphi(t)$, for all $g \in G$. The m coordinates of these orbits are functions of the parameter t. These coordinates can be expressed by $x_i(t)$, for $i = 1, \ldots, m$. The coordinates of the left-invariant vector field associated with φ are obtained as the derivatives $\frac{\mathrm{d}x_i}{\mathrm{d}t}|_{t=0}$, for all $i = 1, \ldots, m$, and are expressed with respect to the basis of vector fields $\left\{\frac{\partial x_i}{\partial t}\right\}_{i=1}^{m}$.

In this paper, we consider a particular subfamily of Lie algebras of the linear algebras. A *linear (*or *matrix) algebra* $\mathfrak{L}$ is that whose vectors are matrices and the Lie bracket is defined by the commutator $[a, b] = a \cdot b - b \cdot a$, for all $a, b \in \mathfrak{L}$. So, every linear Lie algebra is a Lie subalgebra of some general linear algebra $\mathfrak{gl}(\mathbb{C}, n)$, formed by all the $n \times n$ square matrices.

We consider the Lie algebras $\mathfrak{g}_{\mathfrak{n}}$ (with $n \in \mathbb{N} \setminus \{1\}$) associated with the Lie group G_n formed by $n \times n$ upper-triangular matrices with the following structure:

$$g_n(x_{i,j}) = \begin{pmatrix} 1 & x_{1,2} & x_{1,3} & \cdots & x_{1,n-1} & x_{1,n} \\ 0 & 1 & x_{2,3} & \cdots & x_{2,n-1} & x_{2,n} \\ 0 & 0 & 1 & \cdots & x_{3,n-1} & x_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & x_{n-1,n} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}. \tag{1}$$

Then, the Lie algebra $\mathfrak{g}_n$ is a Lie subalgebra of $\mathfrak{gl}(\mathbb{C}, n)$. It was proved in [8] that there exists a basis $\mathcal{B} = \{e_{i,j} \mid 1 \leq i \leq n-1 \wedge i+1 \leq j \leq n\}$ in $\mathfrak{g}_n$. Each vector field $e_{i,j}$ can be expressed with respect to the generator system $\left\{\frac{\partial}{\partial x_{i,j}} \mid 1 \leq i \leq n-1 \wedge i+1 \leq j \leq n\right\}$ of the $\mathcal{C}^\infty(G_n)$-module $\chi(G_n)$ in the following way:

$$e_{k,j} = \frac{\partial}{\partial x_{k,j}} + \sum_{h=1}^{k-1} x_{h,k} \frac{\partial}{\partial x_{h,j}}. \tag{2}$$

The law of the Lie algebra $\mathfrak{g}_n$ with respect to this basis is the following:

$$[e_{i,h}, e_{h,k}] = e_{i,k}, \qquad \begin{cases} i = 1, \ldots, n-1; \\ h = i+1, \ldots, n-1; \\ k = h+1, \ldots, n. \end{cases}$$

As many the vectors of the previous basis $\mathcal{B}$ as the law of the Lie algebra $\mathfrak{g}_n$ with respect to $\mathcal{B}$ will be computed in the algorithm which we show in the next section.

2 The algorithm to compute the law of $\mathfrak{g}_n$

We devote this section to construct and to implement with MAPLE V© the algorithm which allows to compute the law of the Lie algebra $\mathfrak{g}_n$, with $n \in \mathbb{N} \setminus \{1\}$. The only input data needed will be the order n of the matrices of the Lie algebra $\mathfrak{g}_n$.

Step 1. Firstly, it is necessary to determine the dimension of $\mathfrak{g}_n$, which will be used in next steps. This dimension is $d_n = \binom{n}{2}$.

Step 2. The basis $\mathcal{B}$ of the Lie algebra $\mathfrak{g}_n$ is constructed according to the procedure given in the previous section. So, we choose d_n one-parameter subgroups of the Lie group G_n such that their associated vector fields are linearly independent.

Step 3. According to (1), the d_n one-parameter subgroups needed to obtain a basis of the Lie algebra $\mathfrak{g}_n$ are defined as $\varphi_{i,j}(t) = g_n(x_{r,s}(t))$, where:

$$x_{r,s} = \begin{cases} 0, & \text{if } (r,s) \neq (i,j); \\ t, & \text{if } (r,s) = (i,j). \end{cases}$$

Step 4. Once the one-parameter subgroups are chosen, the products $g_n(x_{r,s}) \cdot \varphi_{i,j}(t)$ are computed, for every pair (i,j) where $i,j \in \{1,2,\ldots,n\}$ and $i < j$. For each pair (i,j), the vector field associated with $\varphi_{i,j}$ is obtained by deriving the resulting coordinates in these products with respect to the parameter t in $t = 0$.

Step 5. Finally, all the brackets products are computed starting from the previously obtained basis. In this way, the final output data of the algorithm are the nonzero Lie brackets in the law of $\mathfrak{g}_n$, with respect to the basis calculated before. Note that this basis is an intermediate output in the algorithm.

3 Implementing the algorithm with MAPLE V©

To make easier the implementation of the algorithm with MAPLE V©, we are going to consider it divided into several steps.

Firstly, we start loading the package `linalg` to active the commands related to Linear Algebra. Remind that Lie algebras are vector spaces with a second structure: the Lie-Poisson bracket.

```
> restart:with(linalg):
```

To start running the algorithm, we need to input the order n of the matrices in $\mathfrak{g}_n$. Starting from this data, it is very easy to compute the dimension of $\mathfrak{g}_n$, which is the output denoted by `r`. In this way, Step 1 of the algorithm is finished.

```
> n:=8:
> r:=n*(n-1)/2;
```

In Step 2, we will construct the `r` one-parameter subgroups of G_n. To do it, we define a `r`-dimensional vector `A`. The coordinates of `A` are the complex coordinates in a global chart of the Lie group G_n. The output `r` is used to construct the vector `A`.

```
> A:=matrix(1,r):
  for i from 1 to r do
  A[1,i]:=a.i:
  od:
```

The following cluster constructs the previously mentioned global chart of G_n. So the matrices of G_n are defined starting from the coordinates of `A`, row by row. To do it, we consider a loop which allows us to define each row in the matrix. Inside this loop we program other two loops to define each row in the matrix. The first one defines the elements in the row before the main diagonal and the second one, the elements after the main diagonal. The main diagonal is defined in the sentence `F[j,j]:=1` inside of the main loop of the cluster.

```
> F:=matrix(n,n):
> for j from 1 to n do
  for k from 1 to j-1 do
  F[j,k]:=0:
  od:
  F[j,j]:=1:
  for k from 1 to n-j do
  F[j,j+k]:=A[1,k+sum(n-s,s=1..j-1)]:
  od:
  od:
```

At this point we start to program Step 3. We construct the `r` one-parameter subgroups which will determine later the `r` linearly independent left-invariant vector fields in $\mathfrak{g}_n$. In this way, we define these one-parameter subgroups by means of two nested loops. They will allow us to define the one-parameter subgroups by indicating the row and the column in which the parameter is. In this way, the one-parameter subgroups can be defined and determined. However, other two nested loops are needed inside of the previous two nested loops. These two new loops serve to define all the terms in the $n \times n$ matrix of the one-parameter subgroup. To determine its elements, all of them will be defined as zero, except the located in the main diagonal, which will be defined as ones. Once all the elements in the matrix are defined, we redefine the term determined by the pair `(h,i)` as the parameter `t`.

```
> for h from 1 to n-1 do
  for i from h+1 to n do
     Phi.h.i:=matrix(n,n):
     for j from 1 to n do
     for k from 1 to n do
     Phi.h.i[j,k]:=0:
     od:
     Phi.h.i[j,j]:=1:
     od:
  Phi.h.i[h,i]:=t:
  od:
  od:
```

In Step 4, we program a cluster which computes the left-invariant vector field associated with each one-parameter subgroup defined in the previous step. To do it, we compute the orbits $g_n(x_{r,s}) \cdot \varphi_{i,j}(t)$ for each one-parameter subgroup $\varphi_{i,j}(t)$. These orbits are denoted by `M.h.i` and the associated vector field `Y.h.i` can be obtained by deriving the terms of

`M.h.i` with respect to `t` and by substituting $t = 0$.

```
> for h from 1 to n-1 do
  for i from h+1 to n do
  M.h.i:=evalm(F&*Phi.h.i):
    Y.h.i:=M.h.i:
    for j from 1 to n do
    for k from j to n do
      Y.h.i[j,k]:=unapply(diff(unapply(M.h.i[j,k],t)(t),t),t)(0):
    od:
    od:
  od:
  od:
```

The following cluster reorders the vector fields obtained in the previous one. Instead of depending on two subindices, we search that they depend on a unique subindex. This is got with the following sentences:

```
> for h from 1 to n-1 do
  for i from h+1 to n do
   if h=1 then
    X.(i-1):=Y.h.i:
   else X.(i-h+sum(n-s,s=1..h-1)):=Y.h.i:
   fi:
   od:
   od:
```

Although we have computed a basis of $\mathfrak{g}_n$, these vector fields are expressed as matrices. However, working with the vector expression of these fields is computationally more useful. This is then the objective of the following cluster.

So, we program a cluster which rewrites these vector fields. In this way, each vector field goes on from a $n \times n$ matrix expression to a `r`-dimensional vector expression.

```
> for l from 1 to r do
    Z.l:=matrix(1,r):
    for j from 1 to n-1 do
    for k from j+1 to n do
       if j=1 then
       Z.l[j,k-1]:=X.l[j,k]:
       else Z.l[1,k-j+sum(n-s,s=1..j-1)]:=X.l[j,k]:
       fi:
    od:
    od:
   od:
```

The following loop returns the remaining intermediate outputs: the vector expression of each vector field in the basis of $\mathfrak{g}_n$.

```
> for l from 1 to r do
   print(cat(‘Z‘,l,‘=‘),Z.l);
```

```
   od:
```

In the next cluster the Lie-Poisson bracket is defined. For it, we construct the command `cor`, which will depend on two arguments. These arguments are the subindices of the vector fields multiplied in the bracket, in the the same indicated order.

```
> cor:= proc(i,j)
   local h,producto:
   producto:=array(1..r):
   for h from 1 to r do
   producto[h]:= sum('Z.i[1,k]*diff(Z.j[1,h],a.k)-Z.j[1,k]
       *diff(Z.i[1,h],a.k)','k'=1..r):
   od:
   producto
   end:
```

Finally, we program a cluster which implements Step 5 in the algorithm. This last cluster determines the nonzero brackets in the law of $\mathfrak{g}_n$ and it is based on the following procedure:

For each bracket, the program studies if this is nonzero. To decide this, a variable `m` is considered in the cluster. For each bracket, if a coordinate is nonzero, the variable `m` is increased in one unit. In this way, the nonzero brackets are those which `m` is not equal to zero. Besides, to decide what is the result of such a nonzero bracket, we determine the coordinate equalled to 1 in the bracket. Remember that each vector field of the basis of $\mathfrak{g}_n$ is determined by the coordinate equal 1 (see (2)). In this way, we obtain, as the final outputs, the brackets such that `m` is nonzero; that is, the nonzero brackets in the law of $\mathfrak{g}_n$.

```
> for i from 1 to r-1 do for j from i+1 to r do
   m:=0:
   for k from 1 to r do
   if cor(i,j)[k]<>0 then m:=m+1 fi:
   od:
   if m<>0 then
   for k from 1 to r do
   if cor(i,j)[k]-1=0 then
   print(cat(`[Z`,i,`,Z`,j,`]=Z`,k)):
   fi:
   od:
   fi:
   od:
   od:
```

4 Some final conclusions

The implementation of the algorithm shown in this paper allows us to compute the law of the Lie algebra $\mathfrak{g}_n$ starting from the order n of the matrices in $\mathfrak{g}_n$. Indeed, we obtain all

the nonzero brackets in the law as the final outputs. Besides, we obtain other intermediate outputs: the dimension and a basis of $\mathfrak{g}_n$.

The algorithm have been implemented with MAPLE V© in a computer Pentium IV 2.5 GHz and 256 MB of RAM. In the following table, we show the computational time and the memory used to return the outputs:

Input (order n)	**Dimension of $\mathfrak{g}_n$**	**Computational time**	**Used memory**
2	1	0 s	0 B
3	3	0 s	832 KB
4	6	0.25 s	1.31 MB
5	10	1.3 s	1.44 MB
6	15	7.8 s	1.50 MB
7	21	37.2 s	1.62 MB
8	28	144 s	1.69 MB
9	36	486.1 s	1.87 MB
10	45	1449.7 s	2.00 MB

It can be observed that we have computed the basis of $\mathfrak{g}_n$ for n up to and including 10. Starting from $n = 8$, the time running the computer increases two or three times when the order n increases one unit. So, for $n = 10$, the program runs 24 minutes approximately to compute the law of $\mathfrak{g}_{10}$. However, it is convenient to note that most computational time needed to run the algorithm corresponds to decide if each bracket is nonzero in the law of $\mathfrak{g}_n$.

Note also that in this paper we have only considered matrices of order less than 11, because the Lie groups $\mathfrak{g}_n$ needed to obtain minimal representations of 6-dimensional simply connected nilpotent Lie algebras are those whose order n is less than 7. Besides, to obtain all the conjugacy classes in the Lie groups G_n, we think convenient to translate this problem to the associated Lie algebras. In this way, obtaining an algorithm which allows to compute the law of the Lie algebras is a first step to study such conjugacy classes.

References

[1] R. E. Beck and B. Kolman. Computers in Lie algebras. I. Calculation of inner multiplicities. *SIAM J. Appl. Math.* **25** (1973), 300–312.

[2] J. Draisma. Constructing Lie algebras of first order differential operators. *Journal of Symbolic Computation* **36**:5 (2003), 685–698.

[3] W. A. de Graaf. Classification of Solvable Lie Algebras. *Experimental Mathematics* **14**:1 (2005), 15–25.

[4] W. Fulton and J. Harris. *Representation theory: a first course*. Springer-Verlag, New York, 1991.

[5] N. Jacobson. A note on automorphisms and derivations of Lie algebras, *Proc. Amer. Math. Soc.* **6** (1955), 281–283.

[6] D. Redfern. *The Maple handbook: MAPLE V release 4*. Springer-Verlag, 1996.

[7] M. Postnikov. *Lie groups and Lie algebras. Lectures in Geometry V*, "Nauka", Moscow, 1994.

[8] A. F. Tenorio. Grupos de Lie asociados a álgebras de Lie nilpotentes. Ph.D. thesis. Universidad de Sevilla, 2003.

[9] V. S. Varadarajan. *Lie Groups, Lie Algebras and Their Representations*. Springer, New York, 1984.

[10] W. T. van Est and T. J. Korthagen. Non-enlargeable Lie algebras. *Neederl. Akad. Wetensch. Proc.* **A26** 15-31 (1964).

Juan Carlos Benjumea
Department of Geometry and Topology
University of Seville
jcbenjumea@us.es

Juan Núñez
Department of Geometry and Topology
University of Seville
jnvaldes@us.es

Ángel F. Tenorio
Department of Economics, Quantitative Methods and Economic History
Pablo de Olavide University at Seville
aftenvil@upo.es

Colored partitions and dynamical systems

Farida Benmakrouha Christiane Hespel

Abstract

We study the validation of a family (B_k) of bilinear system, global modelling of an unknown dynamical system (Σ).
Two formal power series in noncommutative variables are used for describing (Σ) : the generating series for the system's behavior (G) and the Chen series for the system's input. The family (B_k) of bilinear systems is described by its rational generatrice series (G_k) such that the coefficients of (G) and (G_k) coincide up to order k.
Computing and bounding these coefficients, we propose an estimation of the error due to approximations by (B_k). This error computation is a sum of differential monomials in the input functions and behavior system. This error computation allows one to better measure the impact of noisy inputs on the convergence of (B_k).

Introduction

The model validation is a crucial problem in system identification[2]. It measures confidence in the model to reproduce the behavior of a dynamic system, under some hypothesis.
In a discret-time approach, the model validation is really an invalidation since it determines wether a discret sample input-output is inconsistent with the model[3].
In [4], the authors develop new methods for validation of continuous-time non-linear systems. They use Barrier certificates whose existence prove inconsistency of a model, from experimental data.
To validate a continuous-time model of an unknown dynamic system [1], we propose an exact symbolic computation of coefficients of rational power series. We use a deterministic model (versus probabilistic one) by considering that data noises are bounded.
So, the purpose of this paper is to apply combinatorial techniques for computing coefficients of rational formal series (G_k) in two noncommutative variables and their differences at order k and k-1.
This in turn may help one to study validation of a family (B_k) of bilinear systems, described by the series (G_k) and global modeling of an unknown dynamical system (Σ).
Computing and bounding these differences, we propose an estimation of the error due to approximations by (B_k). This error computation is a sum of differential monomials in the input functions and behavior system. We identify each differential monomial with its colored multiplicity and analyse our computation in the light of the free differential calculus.
This error computation allows one to better measure the impact of noisy inputs on the

convergence of (B_k). Indeed, one can determine the contribution of the inputs and of the system in the error computation.

1 A local modeling of the unknown system

The problem consists in modeling an unknown dynamic system (Σ) for $t \in [0,T] = \bigcup_{i\in I}[t_i, t_i + d]$, when knowing some correlated sets of input/output.
We construct a behavioral model, based on the identification of its input/output functional (the generating series), in a neighborhood of every t_i, up to a given order k [1, 6]. At once a local modeling by a bilinear system $(B_i)_k$ around every t_i is provided. Then a family $((B_i)_{i\in I})_k$, global modeling of the unknown system is produced, such that the outputs of (Σ) and $((B_i)_{i\in I})_k$ coincide up to order k.

2 The bilinear system

We consider a certain class $(\mathcal{G}P)$ enclosing the electric equation

$$y^{(1)}(t) = f(y(t)) + u(t) \tag{1}$$

where $u(t)$ is the input function
Σ, the unknown system is an affine system.
In this case, equation (1) can be written

$$(\Sigma) \qquad \begin{cases} \dot{x} &= A_0(x) + A_1(x)u(t) \\ y(t) &= x(t) \end{cases}$$

- u(t) is the real input
- x(t) is the current state
- $A_0 = a^{(0)}\frac{d}{dx}$ where $a^{(0)} = f(x)|_{x(0)}$
- $A_1 = \frac{d}{dx}$

The class $(\mathcal{G}P)$ encloses the nonlinear differential equation relating the current excitation i(t) and the voltage v(t) across a capacitor [11]

$$v^{(1)} + k_1 v + k_2 v^2 = i(t)$$

Let $a^{(i)} = f^{(i)}(x)|_{x(0)}$
We notice that the fundamental formula [11]provides the following bilinear system (B_k), approximating at order k :

$$\begin{cases} \dot{x}_k(t) &= (M_0 + M_1 u(t))x_k(t) \\ \overline{y}_k(t) &= \lambda x_k(t) \end{cases}$$

where $\lambda = (x(0) \quad 1 \quad 0 \cdots 0)$

$$x_k(0) = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$M_0 = (C_{z_0 z_1^k})$ (resp $M_1 = (C_{z_1^{k+1}})$) expressed in basis $(C_{z_1^k})$ where C_w is the column of Hankel matrix, indexed by w.

$$M_0 = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ a^{(0)} & a^{(1)} & a^{(2)} & \cdots & a^{(k)} \\ 0 & a^{(0)} & 2a^{(1)} & \cdots & 0 \\ 0 & 0 & a^{(0)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

$$M_1 = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

So, at order k, we obtain the ith derivative of the state vector x as a function of the previous ones. Our solution consists of two steps : to compute $x^{(k-n+i)}_{(k-n)k}(0)$ and to compute the difference of the ith derivative $x^{(i)}_{2k}(0) - x^{(i)}_{2(k-1)}(0)$.

3 First step : Computation of $x^{(k-n+i)}_{(k-n)k}(0)$

By derivating and term's regrouping, we can show that :

$$x^{(k-n+i)}_{(k-n)k}(0)$$

=

$$\sum_{m=1}^{min(i+1,k-1)} a^{(m)} \sum_{l=1}^{k-n-1} \binom{k-n-l+m-1}{m} (a^{(0)} + u^{(0)})^{l-1} x^{(k-n-l+m+i-m)}_{(k-n-l+m)k}(0)$$

+

$$\sum_{m=1}^{i+1} u^{(m)} \sum_{l=1}^{k-n-2} \binom{k-n-l+i}{m} (a^{(0)} + u^{(0)})^{l-1} x^{(k-n-l+i-m)}_{(k-n-l)k}(0) + \delta_{m(i+1)}$$

($\binom{n}{m} = 0 \quad$ if $n < m$)

We analyze now these equations in the light of the free differential calculus. Considering

the derivative $a^{(i)}$ and $u^{(i)}$ specialized in time t=0 as differential letters, it is clear that our computation is a sum of differential monomials in a and u.

3.1 Colored partitions and multiplicities

A number partition or multiplicity is a sequence $\mu = (\mu_1, \mu_2, \mu_3, \cdots)$ (often written as $1^{\mu_1}2^{\mu_2}3^{\mu_3}\cdots$) of nonnegative integers. On a single letter a, the differential monomials become :

$$a^{\mu} = (a^{(i_1)})^{e_1}(a^{(i_2)})^{e_2}\cdots(a^{(i_q)})^{e_q}, \quad 1 \leq i_1 < i_2 < \ldots i_q$$

Such a monomial is indexed by the following partition [12] :

$$\mu = (i_1^{\mu_{i_1}} i_2^{\mu_{i_2}} \cdots i_q^{\mu_{i_q}})$$

Let $C = \{a, u\}$ be a set of two colors. We call colored partition on C an element of the free monoid generated by the cartesian product $N \times N$ i.e. any finite sequence of couples of nonnegative integers

$$\mu = ((\mu_1^a, \mu_1^u), (\mu_2^a, \mu_2^u), \cdots)$$

So, a colored partition μ will denote the differential monomial

$$a^{\mu} = (a^{(i_1)})^{e_1}\cdots(a^{(i_p)})^{e_p}(u^{(j_1)})^{f_1}\cdots(u^{(j_q)})^{f_q}$$

$$1 \leq i_1 < i_2 < \ldots i_p, \quad 1 \leq j_1 < i_2 < \ldots j_q$$

where e_l (resp f_l) $= \mu_{i_l}^a$ (resp $\mu_{j_l}^u$). The weight and the size of μ are defined as follows :

$$wgt(\mu) = \sum_c \sum_k k.\mu_k^c$$

$$size(\mu) = \sum_c \sum_k \mu_k^c$$

The empty partition is noted ϵ.
If L is the set of colored partitions, we define a partial order $\ll$ on L :

$$\nu = \{(\nu_i^a, \nu_i^u)\} \ll \mu = \{(\mu_i^a, \mu_i^u)\}$$

if

$$\nu_i^a \leq \mu_i^a \quad and \quad \nu_i^u \leq \mu_i^u \quad \forall i$$

L, with this partial ordering forms a Young lattice. [13]
We consider now B_i a subset of L defined by :

$$\{\mu / wgt(\mu) = i\}$$

and we note $I(\mu_{max})$ the order ideal generated by μ_{max}, if

$$\mu_{max} = max(\mu / \mu \in B_i)$$

3.2 Combinatorial analysis of our computation

Let us now interpret combinatorially our computation by identifying each differential monomial with its colored multiplicity. The recursive relation is captured by the operation :

$$\mu_{max} \odot c = \sum_{\substack{\nu \in I(\mu_{max}) \\ wgt(\nu)=j \leq i}} c^{(i-j+1)}.\nu$$

By factorizing according to the colored partitions, we get :

$$x_{(k-n)k}^{(k-n+i)} = \sum_{c} \sum_{\substack{\nu \in I(\mu_{max}) \\ wgt(\nu)=j \leq i}} c^{(i-j+1)}.\nu.g^1_{(c^{(i-j+1)}\nu)}$$

where :

$$g^l_{a^{(m)}\nu} = (a^{(0)} + u^{(0)})^{m+1} \sum_{p=m}^{n_l+m} \binom{l}{m} g_v^p$$

and

$$g^l_{u^{(m)}\nu} = (a^{(0)} + u^{(0)})^{m-1} \sum_{p=1}^{n_l} \binom{l+i+1}{m} g_v^p$$

with $n_1 = k - n - 1, \quad n_l = l \quad \forall l > 1$

$$g_\epsilon = 1$$

3.3 Computation of $x_{(k-n)k}^{(k-n+i)}(0)$

We consider now permutations of a colored partition μ on an alphabet $X = \bigcup_{c \in C} X_c$. A permutation [13] of μ is a word in which each letter belongs to X and for each $x_i \in X$, the total number of appearances of x_i in the word is μ_i^c, for some $c \in C$
Let us note $\pi = \xi_1 \xi_2 \cdots \xi_{size(\mu)}$ a permutation of μ and σ_μ the set of permutations of μ.
Since, our alphabet $X_a = \{a^{(p)} | p = 1, min(k-1, i+1)\}$,and $X_u = \{u^{(p)} | p = 1, i+1)\}$ $\xi_j = c^{(i_j)}$, for some c,i_j.
$x_{(k-n)k}^{(k-n+i)}$ is a linear combination of monomial $y_1^{\lambda_1} \cdots y_n^{\lambda_n} (y_i \in X_a \bigcup X_u)$ and all distinct monomials obtained from it by a permutation of variables.
We get finally , if $s = (\sum_j j | \quad \mu_j^u \neq 0)$ and r =size(μ)

$$x_{(k-n)k}^{(k-n+i)} = \sum_{wgt(\mu)=i+1} \mu.(a^{(0)} + u^{(0)})^{k-n+i-r-s} \quad g_\mu^n$$

$$g_\mu^n = \sum_{\pi \in \sigma_\mu} A_1 \prod_{j=2}^{r} A_j + b$$

where:

$$A_j = \begin{cases} \sum_{m_j=i_j}^{m_{j-1}+i_j} \binom{m_j}{i_j} & if \quad \xi_j = a^{(i_j)} \\ \sum_{m_j=1}^{m_{j-1}} \binom{m_j+i-j+2}{i_j} & if \quad \xi_j = u^{(i_j)} \end{cases}$$

$$A_1 = \begin{cases} \sum_{m_1=m}^{k-n-2+m} \binom{m_1}{i_1} & if \quad \xi_1 = a^{(i_1)} \\ \sum_{m_j=1}^{k-n-2} \binom{m_1+i+1}{i_1} & if \quad \xi_1 = u^{(i_1)} \end{cases}$$

and b = 1 if $\xi_1 = u^{(i+1)}$, 0 otherwise.

Remark : $x_{(k-n)k}^{(k-n+i)}$ is not a symmetric polynomial even if its structure is the same, because input and system contributions are different.

4 Second step: Computation of $x_{2k}^{(k+i)}(0) - x_{2(k-1)}^{(k+i)}(0)$

The first derivative coincide up to order k-2, but at order k-1, we have
$x_{2k}^{(k-1)} - x_{2(k-1)}^{(k-1)} = 0$ and $x_{jk}^{(k-1)} - x_{j(k-1)}^{(k-1)} \neq 0$.
Let M (resp P) the set of partitions on the single letter a (resp u)
W_i a subset of M defined by

$$\{\nu | 1 \leq size(\nu) \leq i+2\}$$

V_i a subset of P defined by

$$\{\lambda | size(\lambda) = \lfloor \frac{i}{2} \rfloor , wgt(\lambda) \leq i-2 \quad or \quad \lambda = u^{(i-2)} \quad or \quad \lambda = u^{(i-1)}\}$$

and S_l a subset of L defined by

$$\{\mu | wgt(\mu) = l\}$$

We define now an operation $\nabla : M \times P \times L \mapsto L$

$$\nabla(\nu, \lambda, \mu) = ((\nu_i + \mu_i^a, \lambda_i + \mu_i^c))_i$$

and a subset P_t of $L \quad \forall 0 \leq t \leq i$

$$P_t = \{\tau = \nabla(\nu, \lambda, \mu) | \quad \mu \in S_t, \lambda \in V_i, \nu \in W_i, wgt(\tau) = k+i-1\}$$

We obtain, by a straightforward computation :

$$x_{2k}^{(k+i)} - x_{2(k-1)}^{(k+i)} = \sum_{\substack{\nabla(\nu,\lambda,\mu) \in P_t \\ 0 \leq t \leq i}} \nabla(\nu, \lambda, \mu) \quad h_\nu . f_\lambda . g_\mu^1 . (a^{(0)} + u^{(0)})^{k+i-2-r_1-s}$$

where

$$f_\lambda = \sum_{\pi \in \sigma_\lambda} \prod_{l=1}^{size(\lambda)} \binom{k+i-2l}{k+i-2l-i_j}$$

$$h_\nu = \begin{cases} \sum_{\pi\in\sigma_\nu^1} \prod_{j=1}^{r-2} \binom{i_j + i_{j+1} - 1}{i_{j+1}} \binom{k-2}{i_r} & if \quad size(\nu) \neq 1 \\ 1 & if \quad size(\nu) = 1 \end{cases}$$

with r = size(ν), $r_1 = r + size(\mu)$, s = $(\sum_j j|\mu_j^u \neq 0)$ $\pi = \xi_1\xi_2\cdots\xi_r$, $\xi_j = c^{(i_j)}$, g_μ^1 defined previously.

$$\sigma_\nu^1 = \{\pi \in \sigma_\nu | \pi \neq \nu_1.\mu, size(\nu_1) < size(\nu) \quad and \quad \pi \neq (a^{(1)})^{r-1}.\xi_r\}$$

Taking into account that $\overline{y}_k^{(i)}(0) = x_{2k}^{(i)}(0)$, we obtain a right computation of the output's difference at order k and k-1. By majorization of these output's differences, and when k tends towards infinity, we get an overestimation of the error due to approximation by the (B_k)

5 Conclusion

The validation which is presented in this paper is not statistical. It consists in valuing the convergence of a bilinear models family (B_k) on the unknown system (Σ) by an effective symbolic computation. It displays the respective contributions of the input and of the system itself.
More than a symbolic validation, these computing tools are parameterized by the input and the system's behavior. They can particularly provide a valuation process for rough and oscillating inputs as well as for smooth inputs.

References

[1] F. Benmakrouha, C. Hespel, G. Jacob, E. Monnier *Algebraic Identification algorithm and application to dynamical systems* CASC'2001,The 4th International Workshop on Computer Algebra in Scientific Computing

[2] B.Ninness, G C.Goodwin *Estimation of Model Quality*10th IFAC Symposium on System Identification, Copenhagen July 1994.

[3] Lennart JLjung *The role of model validation for assessing the size of the unmodeled dynamics* IEEE Transactions on automatic control, Vol 42, No 9, September 1997.

[4] Stephen Prajna *Barrier certificates for nonlinear models* Technical report, California Institute of Technology.

[5] A.Juditsky, H.Hjalmarsson, A.Benveniste, B.Delyon, L.Ljung, J.Sjoberg, Q.Zhang, *Nonlinear black-box modeling in system identification:mathematical foundations*, Automatica, 31, 1995.

[6] Benmakrouha F., Hespel C., Jacob G., Monnier E., *A formal validation of Algebraic Identification algorithm: example of Duffing equation*, IMACS ACA'2000, Saint Petersburg, june 25-28, 2000.

[7] Fliess M., *Fonctionnelles causales non linaires et indtermines non commutatives*, Bull. Soc. Math. France 109, pp. 3-40, 1981.

[8] Fliess M., *Sur certaines familles de séries formelles*, Thèse d'état, Université de Paris 7, 1972.

[9] Hespel C., *Une étude des séries formelles non commutatives pour l'Approximation et l'Identification des systèmes dynamiques*, Thèse d'état, Université de Lille 1, 1998.

[10] Benmakrouha F., Hespel C., Monnier E. "Comparison of Identification Methods based on Fuzzy Systems and on Algebraic Model" *2001 WSES International Conference on Fuzzy sets and Fuzzy Systems*, Tenerife, Feb. 2001.

[11] M. Fliess, M.Lamnabhi, F. Lamnahbi-Lagarrigue *An Algebraic approach to nonlinear functional expansions* IEEE Trans. Circuits and Systems, vol. CAS-30, n^0 8,1983, . 554-570.

[12] I.G. Macdonald, *Symmetric Functions and Hall Polynomials*, 2d ed., Oxford Science Publications, 1995.

[13] G.E. Andrews *The theory of Partitions* Encyclopedia of Mathematics and its applications, Addison-Wesleys, 1984

Benmakrouha Farida
Computer Science Department
INSA Rennes
benma@irisa.fr

Hespel Christiane
Computer Science Department
INSA Rennes
hespel@irisa.fr

A Tikhonov-Regularization method for the reconstruction of blurred and noisy images

Abderrahman Bouhamidi Khalid Jbilou

Abstract

In the present paper, we consider a Tikhonov regularization method for large ill-conditioned linear systems. Such a problem arises for example in image restoration where the computation of a stable solution with classical methods is expensive. In some cases the matrix of the linear system may be approximated by Kronecker products and then Tikhonov regularization problem leads to linear matrix equations such as generalized Sylvester equations. We use the global GMRES method which is an orthogonal projection method onto a matrix Krylov subspace to solve the obtained Sylvester equation. Some numerical examples in image restoration are given to illustrate the effectiveness of our approach.

1 Introduction

Consider the linear discrete ill-posed problem

$$\min_x \|Hx - g\|_2, \tag{1}$$

where $H \in \mathbb{R}^{M\times N}$, $x \in \mathbb{R}^N$, $g \in \mathbb{R}^M$ and $M \geq N$. The matrix H is assumed to be very large and of ill-determined rank, i.e., H has some singular values close to the origin.
The right-hand side vector g in (1) represents the output and is assumed to be contaminated by a noise $\mathbf{n}$, i.e., $g = \widehat{g} + \mathbf{n}$.

Such a problem arises, for example, from the discretization of ill-posed problems such as integral equations of the first kind and image restoration [1, 5, 7, 8]. In image restoration, the problem consists of the reconstruction of an original image that has been degraded by a blur and an additive noise. The matrix H represents the blurring matrix, the vector x to be approximated represents the original image, the vector $\mathbf{n}$ represents the additive noise and the vector g represents the blurring and noisy (degraded) image. The well known Tikhonov regularization replaces the problem (1) by the new one

$$\min_x(\|Hx - g\|_2{}^2 + \lambda^2\|Lx\|_2{}^2), \tag{2}$$

where L is a regularization operator chosen to obtain a solution with desirable properties such as small norm or good smoothness. The most popular techniques for determining

the parameter λ are the L-curve criterion [7]; see also [2, 3] and the Generalized Cross-Validation (GCV) method [4, 6]. In image restoration, the problem (2) is generally too large to solve exactly and iterative methods are needed. In Krylov methods, we project (2) onto a Krylov subspace of small dimension and then we solve, at each iteration, this small problem. Another alternative is to project the original problem (1) onto a Krylov subspace and then apply a regularization method to the smaller projected problem. In the present paper, we will consider the case where the matrices H and L are decomposed as Kronecker products. Then the problems (1) and (2) are replaced by new ones involving matrix equations such as the generalized Sylvester equation with small dimensions. To solve these equations we use the global GMRES method introduced in [9].

Let us recall that a Kronecker product of a matrix $A = (a_{ij})$ of size $n \times p$ by a matrix $B = (b_{ij})$ of size $s \times q$ is defined as the $(ns) \times (pq)$ matrix $A \otimes B = (a_{ij}B)$. Some properties of the Kronecker product are given in [10]. The *vec* operator transforms the matrix A to a vector $\mathbf{a}$ of size $np \times 1$ by stacking the columns of A, namely, $\mathbf{a} = vec(A) := (a_{11}, \cdots, a_{1p}, a_{21}, \cdots, a_{2p}, \cdots, a_{n1}, \cdots, a_{np})^T$. For two matrices A and B in $\mathbb{R}^{n\times p}$, we define the following inner product $\langle A, B\rangle_F = tr(A^T B)$ where $tr(Z)$ denotes the trace of the square matrix Z. The well known Frobenius norm denoted by $\| \ . \ \|_F$ is $\| \ A \ \|_F = \sqrt{\langle A, A\rangle_F}$. A system of matrices of $\mathbb{R}^{n\times p}$ is said to be F-orthogonal if it is orthogonal with respect to the scalar product $\langle ., .\rangle_F$.

2 Tikhonov-Sylvester regularization

The Tikhonov regularization method is one of the most popular regularization methods. The parameter λ is chosen to control the "smoothness" of the regularized solution and it is called the regularization parameter. The matrix L defines a (semi)norm on the solution [8] and it is called the regularization operator. Usually, L represents the first or the second discrete derivative operator. The minimizer $\widehat{x}$ of the problem (2) is computed as the solution of the following linear system

$$H_\lambda \widehat{x} = H^T g, \text{ where } H_\lambda = (H^T H + \lambda^2 L^T L). \tag{3}$$

We assume that $H = H_2 \otimes H_1$ and $L = L_2 \otimes L_1$ where H_1, L_1 are matrices of size $n \times n$ and H_2, L_2 are matrices of size $p \times p$. In this case we have $M = N = np$. The problem (3) can be expressed as

$$\big[(H_2 \otimes H_1)^T (H_2 \otimes H_1) + \lambda^2 (L_2 \otimes L_1)^T (L_2 \otimes L_1)\big]\widehat{x} = (H_2 \otimes H_1)^T \mathbf{g}.$$

Using some properties of the Kronecker product, we obtain

$$(H_1^T H_1)\widehat{X}(H_2^T H_2) + \lambda^2 (L_1^T L_1)\widehat{X}(L_2^T L_2) = H_1^T G H_2, \tag{4}$$

where $\widehat{X}$ and G are the matrices of size $n \times p$ such that $vec(\widehat{X}) = \widehat{x}$ and $vec(G) = g$. Setting $A = H_1^T H_1$, $B = L_2^T L_2$, $C = -L_1^T L_1$, $D = H_2^T H_2$, and $E = H_1^T G H_2$, the linear matrix equation (4) is referred to as the generalized Sylvester matrix equation and is written in the following form

$$A\widehat{X}D - \lambda^2 C\widehat{X}B = E. \tag{5}$$

3 The global-GMRES method for generalized Sylvester matrix equations

In this section, we present a numerical Krylov subspace method for solving the generalized Sylvester matrix equation

$$AXD - \lambda^2 CXB = E \tag{6}$$

where $A, C \in \mathbb{R}^{n\times n}$; $B, D \in \mathbb{R}^{p\times p}$; E and $X \in \mathbb{R}^{n\times p}$.

Let $\mathcal{A}_\lambda$ be the linear operator defined from $\mathbb{R}^{n\times p}$ onto $\mathbb{R}^{n\times p}$ as follows

$$\mathcal{A}_\lambda(X) = AXD - \lambda^2 CXB. \tag{7}$$

Then the problem (6) can be written as

$$\mathcal{A}_\lambda(X) = E. \tag{8}$$

Let V be any $n \times p$ matrix and consider the matrix Krylov subspace associated to the pair $(\mathcal{A}_\lambda, V)$ and to an integer k defined by $\mathcal{K}_k(\mathcal{A}_\lambda, V) = span\{V, \mathcal{A}_\lambda(V), \ldots, \mathcal{A}_\lambda^{k-1}(V)\}$. We note that $\mathcal{A}_\lambda^i(V)$ is defined recursively as $\mathcal{A}_\lambda^i(V) = \mathcal{A}_\lambda(\mathcal{A}_\lambda^{i-1}(V))$. Remark that the matrix Krylov subspace $\mathcal{K}_k(\mathcal{A}_\lambda, V)$ is a subspace of $\mathbb{R}^{n\times p}$. The modified global Arnoldi algorithm constructs an F-orthonormal basis $V_1, V_2, \ldots, V_k$ of the matrix Krylov subspace $\mathcal{K}_k(\mathcal{A}_\lambda, V)$, i.e. $\langle V_i, V_j\rangle_F = \delta_{i,j}$ for $i, j = 1, \cdots, k$, where $\delta_{i,j}$ denotes the classical Kronecker symbol. The algorithm is described as follows

ALGORITHM **1** **Modified Global Arnoldi algorithm**

1. `Set` $V_1 = V/\|V\|_F$.

2. `For` $j = 1, \ldots, k.$ `do`

$\tilde{V} = \mathcal{A}_\lambda(V_j)$,

`for` $i = 1, \ldots, j.$ `do`

$h_{i,j} = \langle V_i, \tilde{V}\rangle_F, \quad \tilde{V} = \tilde{V} - h_{i,j}V_i,$

`endfor`

$h_{j+1,j} = \| \tilde{V} \|_F, \quad V_{j+1} = \tilde{V}/h_{j+1,j}.$

`EndFor`.

Hereafter, we need some notations. Let $\mathcal{V}_k$ denote the $n\times kp$ matrix: $\mathcal{V}_k = [V_1, V_2, \ldots, V_k]$. $\tilde{H}_k$ denotes the $(k+1)\times k$ upper Hessenberg matrix whose nonzero entries $h_{i,j}$ are defined by Algorithm 1 and H_k is the $k \times k$ matrix obtained from $\tilde{H}_k$ by deleting its last row. Note that the block matrix $\mathcal{V}_k$ is F-orthonormal which means that the matrices $V_1, \ldots, V_k$ are orthonormal with respect to the scalar product $\langle ., . \rangle_F$. At step k, the approximate solution X_k produced by the global-GMRES method is such that

$$X_k = X_0 + Z_k \text{ where } Z_k \in \mathcal{K}_k(\mathcal{A}_\lambda, R_0), \tag{9}$$

$$R_k = E - \mathcal{A}_\lambda(X_k) \perp_F \mathcal{A}_\lambda\Big(\mathcal{K}_k(\mathcal{A}_\lambda, R_0)\Big), \tag{10}$$

where the notation $\perp_F$ means the orthogonality with respect to the scalar product $\langle ., . \rangle_F$. As can be seen from (9) and (10), we are dealing with an orthogonal projection method

and then we have the minimization property

$$\|R_k\|_F = \min_{X - X_0 \in \mathcal{K}_k(\mathcal{A}_\lambda, R_0)} \|E - \mathcal{A}_\lambda(X)\|_F. \tag{11}$$

Therefore, using the relation (9), the approximation X_k is given by $X_k = X_0 + \mathcal{V}_k(y_k \otimes I_p)$ where y_k is the solution of the following small least-squares problem

$$\min_{y \in \mathbb{R}^k} \| \, \| R_0 \|_F \; e_1 - \tilde{H}_k y \, \|_2, \tag{12}$$

and e_1 is the first unit vector of $\mathbb{R}^{k+1}$.

In the practical implementation, we used the restarted global-GMRES(m) where m is a chosen integer parameter. The global-GMRES(m) algorithm for solving the generalized Sylvester matrix equation (6) is summarized as follows

ALGORITHM **2**. **The global-GMRES(m) for the generalized Sylvester equation**

1. `Choose` X_0, `a tolerance` ε `and set` $k = 0$.
`Compute :` $R_0 = E - \mathcal{A}_\lambda(X_0)$, $\beta = ||R_0||_F$ `and` $V_1 = R_0/\beta$.

2. `Construct the` F`-orthonormal basis` $V_1, V_2, \ldots, V_m$ `by applying Algorithm 1 to the pair` $(\mathcal{A}_\lambda, V_1)$.

3. `Determine` y_m `as the solution of the least squares problem:`

$$\min_{y \in R^m} \| \; \beta \, e_1 - \tilde{H}_m y \, \|_2.$$

`Compute:` $X_m = X_0 + \mathcal{V}_m(y_m \otimes I_p)$ `and the corresponding residual` R_m.

4. `If` $||R_m||_F < \varepsilon$ `Stop;`
`else` $X_0 = X_m$, $R_0 = R_m$, $\beta = ||R_0||_F$, $V_1 = R_0/\beta$, $k = k + 1$, `Goto 2.`

Note that when the operator $\mathcal{A}_\lambda$ is symmetric, the matrix H_m is symmetric and tridiagonal and in this case the computational cost is reduced. The optimal value of the parameter λ is determined by using the L-curve criterion or the GCV method.

4 Projection plus Tikhonv regularization

An economical approach is to project the original problem onto a Krylov subspace and then apply Tikhonov regularization to the small projected minimization problem. Since $H = H_2 \otimes H_1$, the initial minimization problem (1) can be expressed as

$$\min_{X \in X_0 + \mathcal{K}_k(\mathcal{H}, E)} \|G - H_1 X H_2^T\|_F \tag{13}$$

where the linear operator $\mathcal{H}$ is defined by $\mathcal{H}(V) = H_1 V H_2^T$ and V is an $n \times p$ matrix.

To solve (13), we apply the global-GMRES method. Starting from an initial guess X_0 and the corresponding residual R_0, we form an F-orthonormal basis of the matrix Krylov subspace $\mathcal{K}_k(\mathcal{H}, R_0)$ and get the $(k+1) \times k$ matrix $\tilde{H}_k$. The projected problem we wish to solve is

$$\min_y \|\beta e_1 - \tilde{H}_k y\|_2 \tag{14}$$

where $\beta = \|R_0\|_F$ and e_1 is the unit vector of $\mathbb{R}^{k+1}$.

As the matrix $\tilde{H}_k$ may be ill-conditioned, we apply Tikhonov regularization to the projected problem (14). Thus, we consider

$$\min_y(\|\beta e_1 - \tilde{H}_k y\|_2^2 + \lambda_k^2 \|L_k y\|_2^2), \tag{15}$$

where L_k is some chosen regularizing operator. Hence, we solve (15) and we compute the approximation $X_k = X_0 + \mathcal{V}_k(y_{k,\lambda} \otimes I_p)$ where $y_{k,\lambda}$ solves the minimization problem (15). At each outer iteration k, an optimal value $\lambda_{k,opt}$ of the parameter λ is determined by using the GCV method. In our computations, we used the preceding algorithm in a restarted mode. The algorithm is restarted every m iterations where m is a fixed parameter.

5 Numerical examples

In this section we give two examples illustrating our approach for computing the solution of the linear discrete ill-posed problem (2). All computations were carried out using Matlab 6.5 on an Intel Pentium workstation with about 16 significant decimal digits. We illustrate our method in the context of image restoration. The original image is denoted by X in each example and it consists of 256×256 grayscale pixel values in the range $[0, 255]$. Let $\widehat{x}$ be the vector whose entries are the pixel values of the original image X ordered row-wise and let H represent the blurring matrix. Then the vector $\widehat{g} = H\widehat{x}$ represents the associated blurred and noise-free image. We generated a blurred and noisy image $g = \widehat{g} + \mathbf{n}$, where $\mathbf{n}$ is a noise vector with normally distributed random entries with zero mean and with variance chosen such that $||\mathbf{n}||/||g|| = 10^{-2}$. The original image and the noise are not available, but the blurred and noisy image $G = vec(g)$ together with the blurring matrix H are known.

5.1 Example 1

In this example, the original image is the `corridor` image. The blurring matrix H is given by $H = H_2 \otimes H_1 \in \mathbb{R}^{256^2 \times 256^2}$, where $H_1 = H_2 = [h_{ij}]$ and $[h_{ij}]$ is the Toeplitz matrix of dimension 256×256 given by $h_{ij} = \frac{1}{2r-1}$ for $|i-j| \leq r$ and $h_{ij} = 0$ for $|i-j| > r$, with $r = 10$. The blurring matrix H models a uniform blur. In this example, the matrix L in the linear discrete ill-posed problem (2) is $L = L_2 \otimes L_1$, where $L_1 = I_{256}$ and L_2 is the tridiagonal matrix of size 256×256 given by $L_2 = tridiag([1, 2, 1])$. The restored image is obtained by applying the global GMRES(20) (Algorithm 2) to the problem (5). The optimal value $\lambda_{opt} \simeq 0.0014586$ of the parameter λ is computed by using the GCV method.

In this example, the relative error was $\frac{||X - X_{20}||_F}{||X||_F} \simeq 8.1395 \times 10^{-2}$. The results are shown if Figure 1.

5.2 Example 2

In the second example, we use the method described in Section 4. The original image is the `lena`. The blurring matrix H is given by $H = H_2 \otimes H_1 \in \mathbb{R}^{256^2 \times 256^2}$, where $H_1 = I_{256}$ and

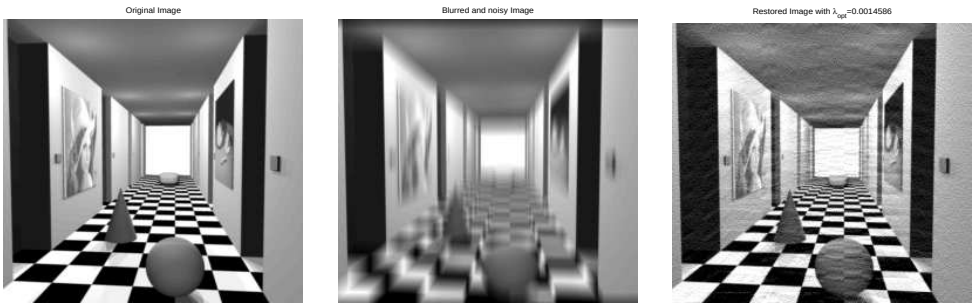


Figure 1: Original image (left), degraded image (center) and restored image (right).

$H_2 = [h_{ij}]$ is the Toeplitz matrix of dimension 256×256 given by $h_{ij} = \frac{1}{2r-1} \exp(-\frac{|i-j|^2}{\sigma^2})$ for $|i-j| \leq r$ and $h_{ij} = 0$ for $|i-j| > r$, with $r = 10$ and $\sigma = 7$. At each outer iteration k, $k = 1, \ldots, 10$, the matrices L_k is the identity matrix. At each outer iteration k, the optimal parameter $\lambda_{k,opt}$ is computed by using the GCV method. The obtained optimal values are $\lambda_{1,opt} = 2.0 \times 10^{-2}$, $\lambda_{2,opt} = 1.2973 \times 10^{-2}$, $\lambda_{3,opt} = 1.1892 \times 10^{-2}$, $\lambda_{4,opt} = 8.8546 \times 10^{-3}$, $\lambda_{5,opt} = 9.6782 \times 10^{-3}$, $\lambda_{6,opt} = 7.0270 \times 10^{-3}$, $\lambda_{7,opt} = 8.1081 \times 10^{-3}$, $\lambda_{8,opt} = 5.9202 \times 10^{-3}$, $\lambda_{9,opt} = 7.0528 \times 10^{-3}$, $\lambda_{10,opt} = 5.2252 \times 10^{-3}$.

The results of this example are given in Figure 2.

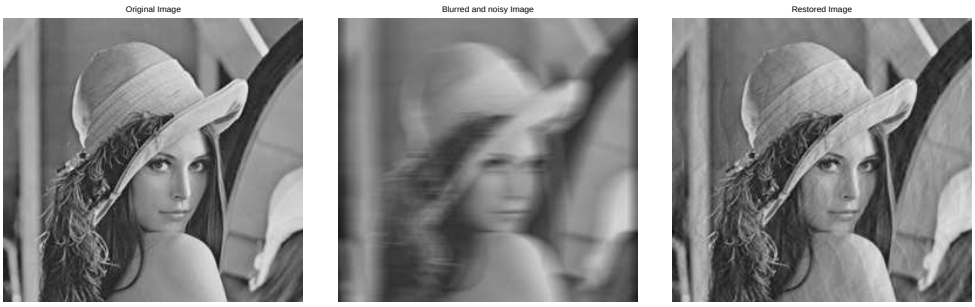


Figure 2: Original image (left), degraded image (center) and restored image (right).

References

[1] C. BREZINSKI, M. REDIVO ZAGLIA, G. RODRIGUEZ AND S. SEATZU, *Extrapolation techniques for ill-conditioned linear systems.*, Numer. Math, 81(1998) pp. 1–29.

[2] D. CALVETTI, G.H. GOLUB AND L. REICHEL, *Estimation of the L-curve via Lanczos bidiagonalization*, BIT, 39(1999) pp. 603–619.

[3] D. Calvetti, B. Lewis and L. Reichel, *GMRES, L-curves and discrete ill-posed problems*, BIT, 42(2002) pp. 44–65.

[4] G.H. Golub, M. Heath, G. Wahba, *Generalized cross-validation as a method for choosing a good ridge parameter,* , Technometrics 21(1979), pp 215–223.

[5] G.H. Golub, P.C. Hansen, D. P. O'Leary, *Tikhonov regularization and total least squares*, SIAM J. Matrix Anal. Appl., 21(1)(1999), pp. 185–194.

[6] G.H. Golub, U. von Matt, *Tikhonov regularization for large scale problems*, in: G.H. Golub, S.H. Lui, F. Luk, R. Plemmons (Eds.), Workshop on Scientific Computing, Springer, New York, 1997, pp. 3–26.

[7] M. Hanke and P.C. Hansen, *Regularization methods for large-scale problems*, Surveys Math. Indust., 3 (1993), pp. 253-315.

[8] P.C. Hansen and D.P. O'Leary, *The use of the L-curve in the regularization of discrete Ill-posed problems*, SIAM J. Sci. Compt., 14(1993), pp. 1487-1503.

[9] K. Jbilou A. Messaoudi H. Sadok, *Global FOM and GMRES algorithms for matrix equations*, Appl. Num. Math., Appl. Num. math., 31(1999), pp. 49–63.

[10] P. Lancaster, L. Rodman, *Algebraic Riccati Equations*, Clarendon Press, Oxford, 1995.

A. Bouhamidi
Laboratoire L.M.P.A
Université du Littoral, 50 rue F. Buisson BP699, F-62228 Calais Cedex, France.
E-mail: bouhamidi@lmpa.univ-littoral.fr;

K. Jbilou
Laboratoire L.M.P.A
Université du Littoral, 50 rue F. Buisson BP699, F-62228 Calais Cedex, France.
E-mail: jbilou@lmpa.univ-littoral.fr;

Well known theorems on triangular systems and the D^5 principle

François Boulier François Lemaire Marc Moreno Maza

Abstract

The theorems that we present in this paper are very important to prove the correctness of triangular decomposition algorithms. The most important of them are not new but their proofs are. We illustrate how they articulate with the D^5 principle.

Introduction

This paper presents the proofs of theorems which constitute the basis of the triangular systems theory: the equidimensionality (or unmixedness) theorem for which we give two formulations (Theorems 1.1 and 1.6) and Lazard's lemma (Theorem 2.1). The first section of this paper is devoted to the proof of the equidimensionality theorem. Our proof is original since it covers in the same time the ideals generated by triangular systems saturated by the set of the initials of the system (i.e. of the form $(A) : I_A^\infty$) and those saturated by the set of the separants of the system (i.e. of the form $(A) : S_A^\infty$). The former type of ideal naturally arises in polynomial problems while the latter one naturally arises in the differential context. Our proof shows also the key role of Macaulay's unmixedness theorem [24, chapter VII, paragraph 8, Theorem 26]. Its importance in the context of triangular systems was first demonstrated by Morrison in [14] and published in [15]. In her papers, Morrison aimed at completing the proof of Lazard's lemma provided in [3, Lemma 2]. Thus Morrison only considered the case of the ideals of the form $(A) : S_A^\infty$, which are the ideals w.r.t. which Lazard's lemma applies. The case of the ideals of the form $(A) : I_A^\infty$ was addressed in [2]. The proof of [2, Theorem 5.1] involves the same gap as that given in [3, Lemma 2]. It was fixed in [1]. The proof provided in [1] does not explicitly use Macaulay's theorem but relies on the properties of regular sequences in Cohen–Macaulay rings, which are the rings in which Macaulay's theorem applies.

What is this gap in the proofs mentioned above ? Among all the indeterminates the elements of a triangular system A depend on, denote $t_1, \ldots, t_m$ the ones which are not main indeterminates. The proofs given in [2, Theorem 5.1] and in [3, Lemma 2] rely implicitly on the assumption that the non zero polynomials which only depend on $t_1, \ldots, t_m$ are not zero divisors modulo the ideal defined by A. This assumption is indeed true but certainly deserves a specific proof.

In the case of the ideals of the form $(A) : I_A^\infty$, let's mention the equidimensionality result of [19] which is not sufficient since it does not solve the problem of the embedded associated

prime ideals of $(A) : I_A^\infty$. In the case of the ideals of the form $(A) : S_A^\infty$, there is a simple proof [16, 4] which unfortunately does not seem to generalize to the ideals of the form $(A) : I_A^\infty$.

The second section of this paper is devoted to the proof of Lazard's lemma. This lemma was communicated by Lazard to the first author a few days before his PhD defense in 1994, with a sketch of proof. The proof given here is very close to the original one. As stated above, Lazard's lemma was first published in [3] but its first complete proof is due to Morrison [14, 15]. Among the few other proofs published afterwards, let's mention the ones given in [18, 4, 8, 17].

In the remaining sections, we show how the equidimensionality theorem and Lazard's lemma apply to the so called "regular chains" [11, 9, 23, 2]. We last recall a few basic algorithms which carry out a generalization of the "D^5" principle [6] for regular chains and which implictly rely on the equidimensionality theorem. Historically, the "D^5" principle suggests to compute modulo zero dimensional ideals presented by triangular systems as if these ideals were prime (whenever a zero divisor is exhibited, the ideal is split). It is its generalization to non zero dimensional ideals which requires the equidimensionality theorem.

Throughout this paper, K denotes a commutative field of characteristic zero.

1 The equidimensionality theorem

In the polynomial ring $R = K[x_1, \ldots, x_n, t_1, \ldots, t_m]$, we consider a polynomial system $A = \{p_1, \ldots, p_n\}$. We assume that $\deg(p_i, x_i) > 0$ and $\deg(p_i, x_k) = 0$ for all $1 \leq i \leq n$ and $i < k \leq n$ i.e. that A is a triangular system w.r.t. at least one ordering such that $x_1 < \cdots < x_n$ and that the x indeterminates are precisely the main indeterminates of the elements of A. The initial of a polynomial p_i is the leading coefficient of p_i, viewed as a univariate polynomial in x_i. The separant of p_i is the polynomial $\partial p_i / \partial x_i$.

In the following, h denotes either the product of the initials of all the elements of A or the product of the separants of all the elements of A.

We are concerned by the properties of the ideal $\mathfrak{A} = (A) : h^\infty$ which is the set of all the polynomials $f \in R$ such that, for some nonnegative integer r and some $\lambda_1, \ldots, \lambda_n \in R$ we have $h^r f = \lambda_1 p_1 + \cdots + \lambda_n p_n$. When h is the product of the initials of the elements of A, the ideal $\mathfrak{A}$ is often denoted $(A) : I_A^\infty$ in the literature. When h is the product of the separants, the ideal $\mathfrak{A}$ is often denoted $(A) : S_A^\infty$.

In general, the ideal $\mathfrak{A}$ may be the trivial ideal R (take $A = \{x_1, x_1 x_2\}$). We assume this is not the case.

Denote $R_0 = K(t_1, \ldots, t_m)[x_1, \ldots, x_n]$ the polynomial ring obtained by "moving the t indeterminates in the base field" of R and $\mathfrak{A}_0$ the ideal $(A) : h^\infty$ in the ring R_0. Denote M the multiplicative family $K[t_1, \ldots, t_m] \setminus \{0\}$ so that $R_0 = M^{-1}R$. Denote $M/\mathfrak{A}$ the image of M by the canonical ring homomorphism $R \to R/\mathfrak{A}$. The elements of $R_0/\mathfrak{A}_0$, which is isomorphic to $(M/\mathfrak{A})^{-1}(R/\mathfrak{A})$, have the form a/b where $a \in R/\mathfrak{A}$ and $b \in M/\mathfrak{A}$. In this section, we prove the following theorem.

Theorem 1.1. *An element $a \in R/\mathfrak{A}$ is zero (respectively regular[1]) if and only if every*

[1] regular = not a zero divisor.

element $a/b \in R_0/\mathfrak{A}_0$ is zero (respectively regular).

Proposition 1.2. *To prove Theorem 1.1, it is sufficient to prove that every element of $M/\mathfrak{A}$ is regular.*

Proof. This is a very classical proposition. If every element of $M/\mathfrak{A}$ is regular then $R_0/\mathfrak{A}_0$, is a subring of the total ring of fractions of $R/\mathfrak{A}$ [24, chapter IV, paragraph 9]. The proposition then follows [24, chapter I, paragraph 19, Corollary 1]. □

Let us recall the Lasker–Noether theorem [24, chapter IV, Theorems 4 and 6].

Theorem 1.3. *(Lasker–Noether theorem)*
In a noetherian ring, every ideal is a finite intersection of primary ideals. Every representation of an ideal $\mathfrak{A}$ as an intersection of primary ideals can be minimized by removing on the one hand the redundant primary ideals and by grouping on the other hand the primary ideals whose intersection is itself primary. The so obtained minimal primary decomposition of $\mathfrak{A}$ is not uniquely defined. However, the number of its components and the radicals of its components (the so called "associated prime ideals" of $\mathfrak{A}$) are uniquely defined.

All the rings considered in this section are noetherian.

Proposition 1.4. *To prove Theorem 1.1, it is sufficient to prove that no associated prime ideal of $\mathfrak{A}$ meets M.*

Proof. According to [24, chapter IV, paragraph 6, Corollary 3], if M does not meet any associated prime ideal of $\mathfrak{A}$ then every element of $M/\mathfrak{A}$ is regular. Theorem 1.1 then follows from Proposition 1.2. □

Recall the definition of the dimension of an ideal.

Definition 1.5. The dimension $\dim \mathfrak{p}$ of a prime ideal $\mathfrak{p}$ of a polynomial ring R with coefficients in a field K is the transcendence degree of the fraction field of $R/\mathfrak{p}$ over K. The dimension $\dim \mathfrak{B}$ of an ideal $\mathfrak{B}$ of R is the maximum of the dimensions of the associated prime ideals of $\mathfrak{B}$.

The rest of this section is completely dedicated to the proof of the following theorem which admits Theorem 1.1 as a corollary. This reformulation of Theorem 1.1 is often convenient for writing proofs.

Theorem 1.6. *The associated prime ideals of $\mathfrak{A}$ have dimension m and do not meet M.*

In order to apply Macaulay's unmixedness theorem, one needs to get rid of the saturation by h. For this, one may use the Rabinowitsch trick [20, section 16.5]. One introduces some new indeterminate x_{n+1} and a new polynomial $p_{n+1} = h\,x_{n+1} - 1$. One denotes A' the triangular system of $R' = R[x_{n+1}]$ obtained by adjoining p_{n+1} to A. One denotes $\mathfrak{A}'$ the ideal (A') of R'. Consider the two following canonical ring homomorphisms:

$$R \xrightarrow{\ \phi\ } h^{-1}\,R \simeq R'/(p_{n+1}) \xleftarrow{\ \pi\ } R'.$$

The isomorphism $h^{-1}R \simeq R'/(p_{n+1})$ is classical [7, Exercise 2.2, page 79]: every element of R corresponds to itself, x_{n+1} corresponds to h^{-1}. If $\mathfrak{B}$ is an ideal of R, one denotes $h^{-1}\mathfrak{B}$ or $(\phi\mathfrak{B})$ the ideal of $h^{-1}R$ generated by $\phi\mathfrak{B}$. If $\mathfrak{B}'$ is an ideal of R' then $\pi\mathfrak{B}'$ is an ideal of $\pi R' = R'/(p_{n+1})$.

Lemma 1.7. *The ideal $\mathfrak{A}'$ is proper. If $\mathfrak{q}'_1 \cap \cdots \cap \mathfrak{q}'_r$ is a minimal primary decomposition of $\mathfrak{A}'$ then $\phi^{-1}(\pi\mathfrak{q}'_1) \cap \cdots \cap \phi^{-1}(\pi\mathfrak{q}'_r)$ is a minimal primary decomposition of $\mathfrak{A}$.*

Proof. We use the notations of extensions and contractions defined in [24, chapter IV, paragraph 8], w.r.t. the ring homomorphism ϕ so that $(\phi\mathfrak{A}) = \mathfrak{A}^e$. The ideal $\pi\mathfrak{A}'$ is equal to the ideal $\mathfrak{A}^e$ since both ideals admit a same generating family: A. By [24, chapter IV, Theorem 15 (a)] we have $\mathfrak{A} = \mathfrak{A}^{ec}$ since $\mathfrak{A} = \mathfrak{A} : h^\infty$. Therefore, since $\mathfrak{A}$ is assumed to be proper, so are $\mathfrak{A}^e$ and $\mathfrak{A}'$.

Consider now a minimal primary decomposition $\mathfrak{q}'_1 \cap \cdots \cap \mathfrak{q}'_r$ of $\mathfrak{A}'$. According to [24, chapter IV, paragraph 5, Remark concerning passage to a residue class ring], $\pi\mathfrak{q}'_1 \cap \cdots \cap \pi\mathfrak{q}'_r$ is a minimal primary decomposition of $\pi\mathfrak{A}' = \mathfrak{A}^e$. Since $\mathfrak{A} = \mathfrak{A}^{ec}$, by [24, chapter IV, Theorem 15 (b) and a comment just above this theorem], the associated prime ideals of $\mathfrak{A}$ do not meet M. By [24, chapter IV, Theorem 17] the intersection $\phi^{-1}(\pi\mathfrak{q}'_1) \cap \cdots \cap \phi^{-1}(\pi\mathfrak{q}'_r)$ is a minimal primary decomposition of $\mathfrak{A}$. □

Proposition 1.8. *To prove Theorem 1.6, it is sufficient to prove that the associated prime ideals of $\mathfrak{A}'$ have dimension m and do not meet M.*

Proof. Let $\mathfrak{p}'$ be an associated prime ideal of $\mathfrak{A}'$ and $\mathfrak{p} = \phi^{-1}(\pi\mathfrak{p}')$ the corresponding associated prime ideal of $\mathfrak{A}$ according to Lemma 1.7. Let a be an element of the subring R of R'. Then $a \in \mathfrak{p}'$ if and only if $a/1 \in \pi\mathfrak{p}'$ and $a/1 \in \pi\mathfrak{p}'$ if and only if $a \in \mathfrak{p}$. Therefore, if $\mathfrak{p}'$ does not meet M then $\mathfrak{p}$ does not either and $\dim \mathfrak{p} \geq m$. If moreover $\dim \mathfrak{p}' = m$ then $x_1, \ldots, x_n$ must depend algebraically on $t_1, \ldots, t_m$ modulo $\mathfrak{p}'$ hence they depend algebraically on $t_1, \ldots, t_m$ modulo $\mathfrak{p}$ and $\dim \mathfrak{p} \leq m$. Combining both inequalities, one concludes that $\dim \mathfrak{p} = m$. □

One distinguishes two sorts of prime ideals associated to an ideal $\mathfrak{A}$: the isolated or minimal ones and the embedded or imbedded ones. An embedded associated prime ideal of $\mathfrak{A}$ is an associated prime of $\mathfrak{A}$ which contains another associated prime ideal of $\mathfrak{A}$. In the context of polynomial rings, its algebraic variety is included (embedded) in that of the associated prime ideal that it contains. One thus sees that, at least in the context of polynomial rings, it is much easier to get informations on the minimal associated prime ideals (they correspond to the irreducible components of the algebraic variety of the ideal [24, chapter VII, paragraph 3, Corollary 3 to Hilbert's Nullstellensatz]) than on the embedded associated prime ideals, which have no such simple geometric meaning (see however [7, section 3.8] for a geometric interpretation of embedded primes). In our case, the problem of the minimal associated prime ideals is easily solved by Lemma 1.10. The problem of the embedded associated prime ideals is solved by a difficult theorem: Macaulay's unmixedness theorem. Recall Krull's principal ideal theorem [24, chapter VII, Theorem 22].

Theorem 1.9. *(principal ideal theorem)*
If a proper ideal $\mathfrak{A}$ of a ring $R = K[x_1, \ldots, x_n]$ admits a generating family formed of k elements ($1 \leq k \leq n$) then $\dim \mathfrak{A} \geq n - k$.

Let us come back to our study of the ideal $\mathfrak{A}'$ of R'.

Lemma 1.10. *The dimension of $\mathfrak{A}'$ is m. Moreover, none of the m–dimensional associated prime ideal of $\mathfrak{A}'$ meets M.*

Proof. Consider an associated prime ideal $\mathfrak{p}'$ of $\mathfrak{A}'$.

First consider the case of h being the product of the initials of the elements of A. Then none of these initials belongs to $\mathfrak{p}'$ (otherwise $\mathfrak{p}'$, which contains $h\,x_{n+1} - 1$ would also contain 1). Thus $x_1, \ldots, x_{n+1}$ are algebraically dependent on $t_1, \ldots, t_m$ over K in $R'/\mathfrak{p}'$ (the polynomials of A' cannot degenerate at all).

Consider now the case of h being the product of the separants of the elements of A. Let $p_\ell = a_d\,x_\ell^d + \cdots + a_1\,x_\ell + a_0$ be any element of A'. Since its separant $s_\ell = d\,a_d\,x_\ell^{d-1} + \cdots + a_1$ does not belong to $\mathfrak{p}'$ (otherwise $\mathfrak{p}'$, which contains $h\,x_{n+1} - 1$ would also contain 1), at least one of the coefficients $a_d, \ldots, a_1$ does not belong to it[2]. Thus $x_1, \ldots, x_{n+1}$ are algebraically dependent on $t_1, \ldots, t_m$ over K in $R'/\mathfrak{p}'$ (the polynomials of A' cannot completely degenerate).

In both cases, $x_1, \ldots, x_{n+1}$ are algebraically dependent on $t_1, \ldots, t_m$ over K in $R'/\mathfrak{p}'$. One then concludes, first that $\dim \mathfrak{p}' \leq m$ hence $\dim \mathfrak{A}' \leq m$, second that if $\dim \mathfrak{p}' = m$ then $\mathfrak{p}' \cap M = \emptyset$. The ideal $\mathfrak{A}'$ admits a basis made of $n+1$ elements in a polynomial ring in $n+m+1$ indeterminates. According to the principal ideal theorem, $\dim \mathfrak{A}' \geq m$. Combining both inequalities, one concludes that $\dim \mathfrak{A}' = m$. □

Let us recall Macaulay's unmixedness theorem [24, chapter VII, Theorem 26].

Theorem 1.11. *(Macaulay's unmixedness theorem)*
If a proper ideal $\mathfrak{A}$ of a polynomial ring $R = K[x_1, \ldots, x_n]$ admits a basis made of k elements ($1 \leq k \leq n$) and if $\dim \mathfrak{A} = n - k$ *then all its associated prime ideals have dimension $n - k$.*

The following proposition, combined to Proposition 1.8, concludes the proof of Theorem 1.6 hence that of Theorem 1.1.

Proposition 1.12. *The associated prime ideals of $\mathfrak{A}'$ have dimension m and do not meet M.*

Proof. The ideal $\mathfrak{A}'$ admits a basis made of $n+1$ elements in a polynomial ring in $n+m+1$ indeterminates. According to Lemma 1.10, its dimension is m. According to Macaulay's unmixedness theorem, all its associated prime ideals have dimension m. According to Lemma 1.10 again, none of these prime ideals meets M. □

Let us state a few easy corollaries to Theorem 1.1.

Corollary 1.13. *The minimal primary decomposition of $\mathfrak{A}$ is uniquely defined.*

[2]The characteristic zero hypothesis is used here.

Proof. By Theorem 1.6, the ideal $\mathfrak{A}$ has no embedded associated prime ideal. The corollary then follows [24, chapter IV, paragraph 5, Theorem 8]. □

Corollary 1.14. *Theorems 1.1 and 1.6 hold if $\mathfrak{A}$ is replaced by any ideal $(A) : S^\infty$ where S is any subset of R containing h, provided that $(A) : S^\infty$ is proper.*

Proof. The ideal $(A) : S^\infty$ is the intersection of the primary components of $(A) : h^\infty$ which do not meet the multiplicative family generated by S. Since the primary components of $(A) : h^\infty$ do not meet M, the primary components of $(A) : S^\infty$ do not meet M either and, Theorem 1.6 holds for this ideal also. Theorem 1.1 follows from Theorem 1.6 and Proposition 1.4. □

Corollary 1.15. *Every regular element of $R_0/\mathfrak{A}_0$ is invertible.*

Proof. Still a well known theorem. By Theorem 1.6, the ideal $\mathfrak{A}_0$ has dimension zero. By [24, chapter VII, paragraph 7], the associated prime ideals of $\mathfrak{A}_0$ are maximal. The ideal $\mathfrak{A}_0$ is thus contained in finitely many prime ideals. There is a bijection [24, chapter III, Theorem 7] between the ideals of R_0 which contain $\mathfrak{A}_0$ and the ideals of $R_0/\mathfrak{A}_0$. This bijection maps prime ideals to prime ideals [24, chapter III, Theorem 11]. The ring $R_0/\mathfrak{A}_0$ thus involves only finitely many prime ideals $\mathfrak{p}_1, \ldots, \mathfrak{p}_r$ which are the associated primes of (0). Assume $a \in R_0/\mathfrak{A}_0$ is regular. By [24, chapter IV, paragraph 6, Corollary 3], the element a belongs to none of the ideals $\mathfrak{p}_1, \ldots, \mathfrak{p}_r$. The ideal generated by a must contain 1 since it would otherwise have associated prime ideals all different from $\mathfrak{p}_1, \ldots, \mathfrak{p}_r$ and there are no such prime ideals. Thus there exists some $\bar{a} \in R_0/\mathfrak{A}_0$ such that $a\,\bar{a} = 1$ and a is invertible. □

Corollary 1.16. *Let $1 \leq i \leq n$ be an index. Denote $A_i = \{p_1, \ldots, p_i\}$. If h is the product of the initials of A, denote h_i the product of the initials of the elements of A_i otherwise, denote h_i the product of the separants of the elements of A_i. Denote $\mathfrak{A}_i = (A_i) : h_i^\infty$. Let $a \in R$ be any polynomial. If a is regular in $R_i/\mathfrak{A}_i$ then a is regular in $R/\mathfrak{A}$.*

Proof. Denote $R_{0,i} = K(t_1, \ldots, t_m)[x_1, \ldots, x_i]$ and $\mathfrak{A}_{0,i} = (A_i) : h_i^\infty$ in $R_{0,i}$. Assume a is regular in $R_i/\mathfrak{A}_i$. Then, by Theorem 1.1 and Corollary 1.15, there exists some $\bar{a} \in R_{0,i}$ such that $a\,\bar{a} - 1 \in \mathfrak{A}_{0,i}$. Since $R_{0,i} \subset R_0$ and $\mathfrak{A}_{0,i} \subset \mathfrak{A}_0$, we have $a\,\bar{a} - 1 \in \mathfrak{A}_0$ and a is invertible in $R_0/\mathfrak{A}_0$. By Theorem 1.1 again, a is regular in $R/\mathfrak{A}$. □

2 Lazard's lemma

In this section, we keep the notations of section 1 but we restrict ourselves to the case of h being the product of the separants of the elements of A. The ideal $\mathfrak{A} = (A) : h^\infty$ is often denoted $(A) : S_A^\infty$ in the literature. It is assumed to be proper.

Theorem 2.1. *(Lazard's lemma)*
The ideal $\mathfrak{A}$ is radical.
The minimal prime ideals of $\mathfrak{A}$ have dimension m and do not meet M.

Before proceeding, let us consider the basic case of a system A made of a single polynomial $p_1 = t_1\,(x_1-1)^3\,(x_1-2)$. Then the separant $h = t_1\,(x_1-1)^2\,(4\,x_1-7)$ involves as a factor the polynomial t_1 which does not depend on x_1 and the multiple factor (x_1-1) of p_1. The ideal $\mathfrak{A}$ is generated by (x_1-2) and satisfies Theorem 2.1. Observe that the theorem would not hold in the case of h being the product of the initials of A only. In that case, the ideal $\mathfrak{A}$, which would be generated by $(x_1-1)^3\,(x_1-2)$, would not be radical.

Proposition 2.2. *To prove Theorem 2.1, it is sufficient to prove that $\mathfrak{A}_0$ is radical.*

Proof. The second statement of Lazard's lemma follows from Theorem 1.6. Let us assume $\mathfrak{A}_0$ is radical. Then $R_0/\mathfrak{A}_0$ does not involve any nilpotent[3] element by [24, chapter IV, Theorem 10 and Corollary]. Thus $R/\mathfrak{A}$ does not either by Theorem 1.1 (for if a is a non zero element of $R/\mathfrak{A}$ then its image $a/1$ in $R_0/\mathfrak{A}_0$ is non zero ; if a power a^d of a were zero, then $(a/1)^d$ would be zero too and $R_0/\mathfrak{A}_0$ would involve nilpotent elements). Therefore, $\mathfrak{A}$ is radical and the proposition is proved. □

In the rest of this section, we prove that $\mathfrak{A}_0$ is radical by proving that $R_0/\mathfrak{A}_0$ is isomorphic to a direct product of fields. Since a direct product of fields does not involve any nilpotent element, the ideal $\mathfrak{A}_0$ is radical and the proof of Lazard's lemma is complete.

Indeed, if $R_1,\ldots,R_k$ are rings then one denotes $S = R_1\times\cdots\times R_k$ their direct product. Elements of S are tuples with k components. Given any two elements $a=(a_1,\ldots,a_k)$ and $b=(b_1,\ldots,b_k)$ of S one defines $a+b$ as $(a_1+b_1,\ldots,a_k+b_k)$ and $a\,b$ as $(a_1\,b_1,\ldots,a_k\,b_k)$. In the ring S, zero is equal to $(0,\ldots,0)$ and one is equal to $(1,\ldots,1)$. If the rings R_i do not involve any nilpotent element then S does not either. This is the case in particular when the rings R_i are fields. See [24, chapter III, paragraph 13] for an equivalent formulation based on direct sums. The following theorem is a generalization of the Chinese Remainder Theorem. See [24, chapter III, paragraph 13, Theorem 32] or [7, Exercise 2.6, page 79].

Theorem 2.3. *(Chinese Remainder Theorem)*
If $\mathfrak{A}_1,\ldots,\mathfrak{A}_k$ are ideals of R such that $\mathfrak{A}_i+\mathfrak{A}_j = R$ whenever $i\neq j$ then the ring $R/(\mathfrak{A}_1\cap\cdots\cap\mathfrak{A}_k)$ is isomorphic to the direct product $(R/\mathfrak{A}_1)\times\cdots\times(R/\mathfrak{A}_k)$.

The proposition below concludes the proof of Theorem 2.1. The scheme of its proof is the original scheme of proof communicated by Daniel Lazard.

Proposition 2.4. *The ring $R_0/\mathfrak{A}_0$ is isomorphic to a direct product of fields.*

Proof. The ring $R_0/\mathfrak{A}_0$ can be constructed incrementally. It is isomorphic to the ring S_n defined by:

$$S_0 = K(t_1,\ldots,t_m),\quad S_i = S_{i-1}[x_i]/(p_i) : s_i^\infty.$$

The proof is an induction on n.

The basis $n=0$ is trivial.

Assume S_{n-1} is a direct product of fields $K_1\times\cdots\times K_r$. Then S_n is isomorphic to the direct product of the rings $K_j[x_n]/(p_n) : s_n^\infty$ for all $1\leq j\leq r$. In the formula above one

[3] A nilpotent element of a ring R is a nonzero element of R a power of which is zero.

assimilates the polynomials p_n and s_n with their images by the canonical ring homomorphisms, noticing that the image of the separant of p_n in each $K_j[x_n]$ is the separant of the image of p_n in this ring.

Therefore, in each $K_j[x_n]$, the ideal $(p_n) : s_n^\infty$ is generated by the product of the simple irreducible factors of p_n. It is thus the intersection of the maximal ideals $\mathfrak{m}_\ell$ generated by these factors. According to the Chinese Remainder Theorem, each $K_j[x_n]/(p_n) : s_n^\infty$ is isomorphic to the direct product of the fields $K_j[x_n]/\mathfrak{m}_\ell$. Since direct products are associative, the ring S_n itself is a direct product of fields. □

Corollary 2.5. *Theorem 2.1 holds if $\mathfrak{A}$ is replaced by any ideal $(A) : S^\infty$ where S is any subset of R containing the separants of the elements of A, provided that $(A) : S^\infty$ is proper.*

Proof. The ideal $(A) : S^\infty$ is the intersection of the primary components of $\mathfrak{A}$ which do not meet the multiplicative family generated by S. Since $\mathfrak{A}$ is radical, its primary components are prime ideals [24, chapter IV, Theorem 5]. Thus the primary components of $(A) : S^\infty$ are prime ideals and $(A) : S^\infty$ is radical. The dimension properties shared by all the associated prime ideals of $\mathfrak{A}$ also hold for all the associated prime ideals of $(A) : S^\infty$. □

3 Regular chains

We consider the polynomial ring $R = K[x_1, \ldots, x_n, t_1, \ldots, t_m]$. We assume that the $m + n$ indeterminates are ordered according to some total ordering $\mathscr{O}$. Let p be any polynomial of $R \setminus K$. The greatest indeterminate w.r.t. $\mathscr{O}$ among the indeterminates p depends on is called the *main indeterminate* of p. We consider a *triangular* system $A = \{p_1, \ldots, p_n\}$ of R i.e. a polynomial system whose elements have distinct main indeterminates. Renaming the indeterminates if necessary, we assume that the main indeterminate of p_i is x_i for each $1 \leq i \leq n$. The multiplicative family M, the initials and the separants of the elements of A are then defined as in section 1.

Fix some $1 \leq i \leq n$. Denote A_i the system $\{p_1, \ldots, p_i\}$. Denote h_i the product of the initials of the elements of A_i. Denote R_i the ring $K[t_1, \ldots, t_m, x_1, \ldots, x_i]$. Denote $R_{0,i}$ the ring $K(t_1, \ldots, t_m)[x_1, \ldots, x_i]$. Denote $\mathfrak{A}_i$ the ideal $(A_i) : h_i^\infty$ of R and $\mathfrak{A}_{0,i}$ the ideal $(A_i) : h_i^\infty$ of $R_{0,i}$. Denote $R_0 = R_{0,n}$ and $\mathfrak{A} = \mathfrak{A}_n$.

Definition 3.1. The system A is a *regular chain* if, for each $2 \leq i \leq n$, the initial of p_i is regular in the ring $R_{i-1}/\mathfrak{A}_{i-1}$. Assume A is a regular chain. Then A is said to be *squarefree* if, for each $1 \leq i \leq n$, the separant of p_i is regular in $R_i/\mathfrak{A}_i$.

The above definition is not exactly the same as that of [2, Definition 4.1] but they are strictly equivalent. The difference is that, in [2], the t indeterminates greater than x_i would have been withdrawn from the rings R_{i-1} and R_i. This change is not important for the elements of A_i do not depend on the t indeterminates greater than x_i and, by [24, chapter I, paragraph 16, Theorem 6], if $\bar{R}$ is a ring, a is one of its elements and x is an indeterminate over it then a is zero (respectively regular) if and only if it is zero (respectively regular) in the ring $\bar{R}[x]$. The following results are corollaries to Theorems 1.1 and 2.1.

Corollary 3.2. *The system A is a* regular chain *if, for each $2 \leq i \leq n$, the initial of p_i is invertible in the ring $R_{0,i-1}/\mathfrak{A}_{0,i-1}$. Assume A is a regular chain. Then A is squarefree if, for each $1 \leq i \leq n$, the separant of p_i is invertible in $R_{0,i}/\mathfrak{A}_{0,i}$.*

Proof. It is an immediate corollary of Theorem 1.1 (enlarging the set of the t indeterminates with the x indeterminates which are not needed). □

Corollary 3.3. *Assume A is a squarefree regular chain. Then $\mathfrak{A}$ is radical. Its minimal prime ideals have dimension m and do not meet M.*

Proof. By Corollary 1.16 and the definition of squarefreeness, the separants of the elements of A are regular in $R/\mathfrak{A}$. Thus they do not lie in any associated prime ideal of $\mathfrak{A}$ by [24, chapter IV, paragraph 6, Corollary 3]. Thus, denoting S_A^∞ the multiplicative family that they generate, $\mathfrak{A} = \mathfrak{A} : S_A^\infty$ and the proof follows from Corollary 2.5. □

3.1 Splittings

In this section, we provide two propositions which permit to justify many algorithms carrying out the "D^5" principle for triangular systems [6]. We keep the notations of section 3 and we assume that A is a regular chain. Let $1 \leq i \leq n$ be an index. Assume that there exists a factorization $p_i = b\,c$ in $(R_{0,i-1}/\mathfrak{A}_{0,i-1})[x_i]$ such that $0 < \deg(b, x_i)$, $\deg(c,\, x_i) < \deg(p_i, x_i)$. For each $1 \leq j \leq n$, denote $B_j = A_j$ if $j < i$ otherwise denote $B_j = (A_j \setminus \{p_i\}) \cup \{b\}$. Denote $B = B_n$. For each $1 \leq j \leq n$, denote $h_{b,j}$ the product of the initials of the elements of B_j and $\mathfrak{B}_j$ the ideal $(B_j) : h_{b,j}^\infty$ of R and $\mathfrak{B}_{0,j}$ the ideal $(B_j) : h_{b,j}^\infty$ of $R_{0,j}$. Replacing b by c in the formulas, define C and for each $1 \leq j \leq n$, define C_j, $h_{c,j}$, $\mathfrak{C}_j$ and $\mathfrak{C}_{0,j}$ Denote $\mathfrak{B}_0 = \mathfrak{B}_{0,n}$ and $\mathfrak{C}_0 = \mathfrak{C}_{0,n}$.

Proposition 3.4. *The triangular sets B and C are regular chains. For each $1 \leq j \leq n$ we have $\mathfrak{A}_j \subset \mathfrak{B}_j$ and $\mathfrak{A}_j \subset \mathfrak{C}_j$.*

Proof. We focus on the set B. The arguments for C are similar. Since $0 < \deg(b, x_i) < \deg(p_i, x_i)$, the set B is triangular. For each $1 \leq j < i$ we have $A_j = B_j$ thus $\mathfrak{A}_j = \mathfrak{B}_j$ and B is a regular chain up to index $i-1$. In the ring $R_{0,i-1}$, the initial of p_i is the product of the initials of b and c. Since A is a regular chain, the initial of p_i is invertible in $R_{0,i-1}/\mathfrak{A}_{0,i-1}$. Therefore, the initial of b is invertible in $R_{0,i-1}/\mathfrak{B}_{0,i-1}$. By Corollary 3.2, the set B is a regular chain up to index i and $\mathfrak{A}_i \subset \mathfrak{B}_i$. Let $i < j \leq n$ be an index. We have $\mathfrak{A}_j \subset \mathfrak{B}_j$. Thus the initial of p_j, which is invertible in $R_{0,j-1}/\mathfrak{A}_{0,j-1}$, is also invertible in $R_{0,j-1}/\mathfrak{B}_{0,j-1}$. By Corollary 3.2, the set B is a regular chain up to any index and $\mathfrak{A}_j \subset \mathfrak{B}_j$. □

We have proved that $\mathfrak{A} \subset \mathfrak{B} \cap \mathfrak{C}$. In general the equality does not hold because of possible common factors of b and c. In the particular case of squarefree regular chains, b and c have no common factors and the equality holds, the following proposition shows.

Proposition 3.5. *Assume A is squarefree. Then so are B, C and we have $\mathfrak{A} = \mathfrak{B} \cap \mathfrak{C}$. Moreover, the sets of the minimal prime ideals of $\mathfrak{B}$ and $\mathfrak{C}$ form a partition of the set of the minimal prime ideals of $\mathfrak{A}$.*

Proof. First we prove that B and C are squarefree regular chains. As in the above proof, we focus on B. By Proposition 3.4, the set B is a regular chain. Assume A is squarefree. Let $1 \leq j < i$ be an index. Since $A_j = B_j$, the separant of p_j is regular in $R_j/\mathfrak{B}_j$ and B is a squarefree regular chain up to index $i-1$. Denote s_i, s_b and s_c the separants of p_i, b and c. We have $s_i = s_b\, c + s_c\, b$. Let us prove that s_b is regular in $R_i/\mathfrak{B}_i$. Since A is squarefree, s_i is invertible in $R_{0,i}/\mathfrak{A}_{0,i}$. By Proposition 3.4, for each $1 \leq j \leq n$ we have $\mathfrak{A}_j \subset \mathfrak{B}_j$ thus s_i is also invertible in $R_{0,i}/\mathfrak{B}_{0,i}$. Then, using the fact that $b \in B_i$ we see that $s_b\, c$, hence s_b, is invertible in $R_{0,i}/\mathfrak{B}_{0,i}$. By Corollary 3.2, the set B is a squarefree regular chain up to index i. Let $i < j \leq n$ be an index. Using again the fact that $\mathfrak{A}_j \subset \mathfrak{B}_j$, we see that the separant of p_j is regular in $R_j/\mathfrak{B}_j$. Thus B is a squarefree regular chain up to any index.

Similar statements prove that C is a squarefree regular chain.

By Corollary 3.3, the ideals $\mathfrak{B}$ and $\mathfrak{C}$ are radical. They are equal to the intersections of their minimal prime ideals by [24, chapter IV, Theorem 5]. To conclude the proof of the proposition, it is thus sufficient to prove that the sets of the minimal prime ideals of $\mathfrak{B}$ and $\mathfrak{C}$ form a partition of the set of the minimal prime ideals of $\mathfrak{A}$. Denote V, V_b and V_c the sets of zeros of $\mathfrak{A}_0$, $\mathfrak{B}_0$ and $\mathfrak{C}_0$ in the algebraic closure of $K(t_1, \ldots, t_m)$. Since these ideals have dimension zero, these sets are finite. The minimal prime ideals of $\mathfrak{A}$, $\mathfrak{B}$ and $\mathfrak{C}$ have dimension m and do not meet M. Therefore, by [24, chapter IV, Theorem 15(d)], the ring homomorphism $R \to R_0$ provides a bijection between the minimal prime ideals of $\mathfrak{A}$ (respectively $\mathfrak{B}$, $\mathfrak{C}$) and those of $\mathfrak{A}_0$ (respectively $\mathfrak{B}_0$, $\mathfrak{C}_0$) hence, using [24, chapter VII, paragraph 3, Corollary 2], a bijection between the minimal prime ideals of $\mathfrak{A}$ (respectively $\mathfrak{B}$, $\mathfrak{C}$) and the elements of V (respectively V_b, V_c). It is thus sufficient to prove that V_b and V_c form a partition of V. The cardinal $|V|$ of V is the product $\prod_{j=1}^{n} \deg(p_j, x_j)$. Similar statements hold for V_b and V_c. Since $\deg(p_i, x_i) = \deg(b, x_i) + \deg(c, x_i)$ we see first that $|V| = |V_b| + |V_c|$. Second, we have $V_b \subset V$ and $V_c \subset V$. Third, $V_b \cap V_c$ is empty for a common zero of $\mathfrak{B}$ and $\mathfrak{C}$ would annihilate $s_i = s_b\, c + s_c\, b$ which is invertible. Therefore $V = V_b \cup V_c$, the sets V_b and V_c form a partition of V and the proposition is proved. □

3.2 The D^5 principle for triangular systems

In this section, we provide the scheme of many algorithms carrying out the "D^5" principle for triangular systems. More efficient algorithms can be found in [13, 12]. See also [22, 21, 5]. The triangular set A is assumed to be a regular chain.

Definition 3.6. For every $a \in R$ we define the pseudoremainder of a by A as

$$\mathrm{prem}(a,\, A) \underset{\text{def}}{=} \mathrm{prem}(\ldots \mathrm{prem}(\mathrm{prem}(a,\, p_n,\, x_n),\, p_{n-1},\, x_{n-1}) \ldots,\, p_1,\, x_1).$$

The pseudoremainder algorithm is based on [24, chapter I, paragraph 16, Theorem 9]. It is defined in [10, volume 2, page 407]. The next proposition is proved in [2, Theorem 6.1].

Proposition 3.7. *For every $a \in R$ we have $a \in \mathfrak{A}$ if and only if* $\mathrm{prem}(a,\, A) = 0$.

The parameter a of *algebraic_inverse* denotes an element of R. The function returns an inverse of a in $R_0/\mathfrak{A}_0$ or fails. If it succeeds then a is proved invertible in $R_0/\mathfrak{A}_0$ hence regular

in $R/\mathfrak{A}$ by Theorem 1.1. The function thus implicitly relies on the equidimensionnality theorem. If it fails by encountering a zero divisor, it exhibits a nontrivial factorization of some element p_i of A. The exhibited factorization might allow some calling function to split A as two regular chains by using Proposition 3.4. Observe that the function may fail even if a is regular in $R/\mathfrak{A}$ for it checks the regularity of many different elements of $R/\mathfrak{A}$.

```
function algebraic_inverse (a, A)
begin
  if a ∈ K(t_1, ..., t_m) then
    if a ≠ 0 then
      1/a
    else
      the inverse computation fails (inversion of zero)
    endif
  else
    let x_i be the main indeterminate of a
    (u_1, u_2, u_3) := extended_Euclid (a, p_i, x_i, A)
    if u_3 ≠ 1 then
      the inverse computation fails (inversion of a zero divisor): u_3 is a factor of p_i
    else
      u_1
    endif
  endif
end
```

Here is the generalization of the extended Euclidean algorithm called by *algebraic_inverse*. The main indeterminate of the two polynomials a and b is x_i. The polynomials a and b are viewed as polynomials in $(R_{0,i-1}/\mathfrak{A}_{0,i-1})[x_i]$. The function fails or returns a triple $U = (u_1, u_2, u_3)$ of elements of $(R_{0,i-1}/\mathfrak{A}_{0,i-1})[x_i]$ satisfying a Bézout identity in the ring $(R_{0,i-1}/\mathfrak{A}_{0,i-1})[x_i]$ i.e. a relation $u_1\, a + u_2\, b = u_3$. A proof that U satisfies a Bézout identity can be designed by using the two following loop invariants (i.e. properties which hold each time the loop condition is evaluated). These loop invariants are natural generalizations of the very classical loop invariants of the basic extended Euclidean algorithm:

- $u_1\, a + u_2\, b = u_3$ and $v_1\, a + v_2\, b = v_3$ in $(R_{0,i-1}/\mathfrak{A}_{0,i-1})[x_i]$;
- the set of the common divisors of u_3 and v_3 is equal to the set of the common divisors of a and b.

Observe that the second invariant is stated without using the word "gcd" which would be controversial in this context for the ring $(R_{0,i-1}/\mathfrak{A}_{0,i-1})[x_i]$ is not a UFD. A definition of the gcd in this context is however provided in [13]. Observe that the function needs to recognize zero in $R_{0,i-1}/\mathfrak{A}_{0,i-1}$ in order to evaluate the loop condition and to determine the degree of u_3 after the loop execution. This is achieved by Proposition 3.7. The function also

needs to check the regularity of the leading coefficient of v_3 before performing the Euclidean division. This can be achieved using *algebraic_inverse*.

function *extended_Euclid* (a, b, x_i, A)
begin
  $U := (1, 0, a)$
  $V := (0, 1, b)$
  while $v_3 \neq 0$ do
    $q :=$ the quotient of the Euclidean division of u_3 by v_3 in $(R_{0,i-1}/\mathfrak{A}_{0,i-1})[x_i]$
    $T := V$
    $V := U - q\,V$
    $U := T$
  done
  $c :=$ the coefficient of $x_i^{\deg(u_3, x_i)}$ in u_3
  return *algebraic_inverse* $(c, A)\,U$
end

References

[1] Philippe Aubry. *Ensembles triangulaires de polynômes et résolution de systèmes algébriques. Implantation en Axiom.* PhD thesis, Univ. Paris VI, 1999.

[2] Philippe Aubry, Daniel Lazard, and Marc Moreno Maza. On the theories of triangular sets. *Journal of Symbolic Computation*, 28:105–124, 1999.

[3] François Boulier, Daniel Lazard, François Ollivier, and Michel Petitot. Representation for the radical of a finitely generated differential ideal. In *proceedings of ISSAC'95*, pages 158–166, Montréal, Canada, 1995.

[4] François Boulier, Daniel Lazard, François Ollivier, and Michel Petitot. Computing representations for radicals of finitely generated differential ideals. Technical report, Université Lille I, LIFL, 59655, Villeneuve d'Ascq, France, 1997. (ref. IT306, december 1998 version published in the habilitation thesis of Michel Petitot).

[5] Driss Bouziane, Abdelillah Kandri Rody, and Hamid Maârouf. Unmixed–Dimensional Decomposition of a Finitely Generated Perfect Differential Ideal. *Journal of Symbolic Computation*, 31:631–649, 2001.

[6] Jean Della Dora, Claire Dicrescenzo, and Dominique Duval. About a new method for computing in algebraic number fields. In *Proceedings of EUROCAL85, vol. 2*, volume 204 of *Lecture Notes in Computer Science*, pages 289–290. Springer Verlag, 1985.

[7] David Eisenbud. *Commutative Algebra with a View Toward Algebraic Geometry*, volume 150 of *Graduate Texts in Mathematics*. Springer Verlag, 1995.

[8] Évelyne Hubert. Factorization free decomposition algorithms in differential algebra. *Journal of Symbolic Computation*, 29(4,5):641–662, 2000.

[9] Mickael Kalkbrener. A Generalized Euclidean Algorithm for Computing Triangular Representations of Algebraic Varieties. *Journal of Symbolic Computation*, 15:143–167, 1993.

[10] Donald Erwin Knuth. *The art of computer programming*. Addison–Wesley, 1966. Second edition.

[11] Daniel Lazard. A new method for solving algebraic systems of positive dimension. *Discrete Applied Mathematics*, 33:147–160, 1991.

[12] Marc Moreno Maza. On Triangular Decompositions of Algebraic Varieties. Technical report, NAG, 2000. (presented at the MEGA2000 conference, submitted to the JSC).

[13] Marc Moreno Maza and Renaud Rioboo. Polynomial gcd computations over towers of algebraic extensions. In *Proceedings of AAECC11*, pages 365–382. Springer Verlag, 1995.

[14] Sally Morrison. Yet another proof of Lazard's lemma. private communication, december 1995.

[15] Sally Morrison. The Differential Ideal $[P] : M^\infty$. *Journal of Symbolic Computation*, 28:631–656, 1999.

[16] François Ollivier. A proof of Lazard's lemma. private communication, october 1998.

[17] Brahim Sadik. Une note sur les algorithmes de décomposition en algèbre différentielle. *Comptes Rendus de l'Académie des Sciences*, 330:641–646, 2000.

[18] Josef Schicho and Ziming Li. A construction of radical ideals in polynomial algebra. Technical report, RISC, Johannes Kepler University, Linz, Austria, august 1995.

[19] Shang–Ching Chou and Xiao–Shan Gao. On the dimension of an arbitrary ascending chain. *Chinese Bulletin of Science*, 38:799–904, 1993.

[20] Bruno Louis van der Waerden. *Algebra*. Springer Verlag, Berlin, seventh edition, 1966.

[21] Dongming Wang. Decomposing polynomial systems into simple systems. *Journal of Symbolic Computation*, 25:295–314, 1998.

[22] Wu Wen Tsün. On the foundation of algebraic differential geometry. *Mechanization of Mathematics, research preprints*, 3, 1987.

[23] L. Yang and J. Zhang. Searching dependency between algebraic equations: an algorithm applied to automated reasoning. *Artificial Intelligence in Mathematics*, pages 147–156, 1994.

[24] Oscar Zariski and Pierre Samuel. *Commutative Algebra*. Van Nostrand, New York, 1958.

François Boulier
LIFL, Université Lille I, 59655 Villeneuve d'Ascq, France
boulier@lifl.fr, http://www.lifl.fr/~boulier

François Lemaire
LIFL, Université Lille I, 59655 Villeneuve d'Ascq, France
lemaire@lifl.fr, http://www.lifl.fr/~lemaire

Marc Moreno Maza
ORCCA, University of Western Ontario, London, N6A 5B7 Canada
moreno@orcca.on.ca, http://www.csd.uwo.ca/~moreno

Unique Normal Forms for Nilpotent Vector Fields of Higher Dimensions *

Guoting Chen

Abstract

Unique normal forms for nilpotent planar vector fields have been studied under the graded Lie algebra context. A unique normal form with respect to the classical degree up to certain order is given for nilpotent vector fields in dimensions 3 and 4. New orderings are used and the corresponding unique normal forms are given up to certain order. Their comparison leads to the important conclusion that the unique normal form depends on the ordering and the choices of complementary subspaces. Maple implementations are used to study the problem and to compute the bases of some complementary subspaces.

Introduction

Normal forms of dynamical systems or vector fields are basic and important tools in bifurcation theory of vector fields, and this has a long history. Recently further reduction of classical normal forms has been studied in many ways in order to obtain unique normal forms and some general methods are given ([2, 5, 9, 10, 13]). But the problem of the unique (simplest) normal form for a given vector field is far from trivial. For non nilpotent vector fields, unique normal forms are known in many cases in $\mathbf{R}^2$ and $\mathbf{R}^3$ (for example in [2, 6, 9, 14, 15]). For the nilpotent cases the question is more difficult. In dimension 2 it is first considered in [3] and [10]. A special case is studied in [6]. Other studies are done in [4, 14].

In the present paper we study unique normal forms for formal vector fields in dimensions 3 and 4 with a nilpotent linear part of index one. For the dimensional 3 case a unique normal form up to terms of degree 5 is given in [5], and up to terms of degree 8 is given in [16]. For the dimensional 4 case, a unique normal form up to terms of degree 3 is given in [5].

Normal form theory in a graded Lie algebra context is given in [2]. New grading for vector fields has been introduced in [10] and has been used successifuly for nilpotent planar vector fields of dimension 2 in [3, 6, 10, 14]. Although in dimension 2 it seems that new grading unique normal forms induce unique normal forms in the classical order, one can ask the question: is it necessarily true in higher dimensional case? This question has never been raised. In fact it seems that it is usually admitted. Our results give a negative answer to this question.

*Dedicated to Jean Della Dora on the occasion of his 60th birthday

We first consider vector fields in $\mathbf{R}^3$ with an equilibrium at the origin with a nilpotent linear part of index one $X_1 = y\partial_x + z\partial_y$, where $\partial_x = \partial/\partial x$, $\partial_y = \partial/\partial y$ and $\partial_z = \partial/\partial z$. We shall write $X(x, y, z) = \sum_{k=1}^{+\infty} X_k$, where X_k denotes the homogeneous part of degree k. We use also notations for the corresponding dynamical system

$$\dot{x} = y + \cdots, \qquad \dot{y} = z, \qquad \dot{z} = \cdots$$

where the dots represent terms of degree ≥ 2. The matrix of the linear part is the nilpotent matrix of index one

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

In Section 1, we give a brief presentation of the method that we shall follow, that is the graded Lie algebra and normal forms using multiple Lie brackets (see [2, 7]). In Section 2 we compute the structure of unique normal form for nilpotent vector fields in $\mathbf{R}^3$ with respect to the classical order (total degree). In Section 3, we consider a linear grading function (see [10]) for vector fields in $\mathbf{R}^3$ and compute unique normal forms with respect to the new grading by showing two different choices of complementary subspaces. In Section 4, some comparisons are done with nilpotent vector fields in dimensions 4 where a unique normal form is given up to certain order.

1 Preliminaries on unique normal forms in a graded Lie algebra

We give a brief review of Baider's method (see [2, 7]) for unique normal forms that we shall follow in this paper. Notice that one can also use the method of [10].

Let $\{\mathcal{L}_j\}$ be a family of vector spaces over $\mathbf{R}$, let $\oplus\mathcal{L}_j$ denote their direct sum, and let $\mathcal{L} = \prod_{j\geq 0} \mathcal{L}_j$ denote their complete direct sum. Elements of $\mathcal{L}$ are treated as (formal) infinite series $\ell = \ell_0 + \ell_1 + \cdots$ with $\ell_j \in \mathcal{L}_j$. Elements of the form $\ell = 0 + \cdots + 0 + \ell_j + 0 + \cdots$ are homogeneous (of order j), and in such cases we simply write $\ell = \ell_j$, thereby viewing $\mathcal{L}_j$ as a subspace of $\mathcal{L}$. Subscript always indicates homogeneity of the corresponding order.

Throughout the section we assume $\mathcal{L} = \prod_{j\geq 0} \mathcal{L}_j$ is a graded Lie algebra, *i.e.* a Lie algebra over $\mathbf{R}$ with bracket $[\ ,\ ]$ satisfying $[\mathcal{L}_i, \mathcal{L}_j] \subset \mathcal{L}_{i+j}$.

For any $m \in \mathcal{L}$ the adjoint mapping $\mathrm{ad}(m) : \mathcal{L} \to \mathcal{L}$ is defined by $\ell \to [m, \ell]$. Note that if $m \in \mathcal{L}_i$ then $\mathrm{ad}(m)|_{\mathcal{L}_j} : \mathcal{L}_j \to \mathcal{L}_{i+j}$. We have

$$\exp(\mathrm{ad}(m))\ell = \sum_{j\geq 0} \frac{1}{j!} (\mathrm{ad}(m))^j \ell.$$

For each $k \geq 0$ define the k-jet of $\ell = \ell_0 + \ell_1 + \cdots \in \mathcal{L}$ to be $J_k(\ell) = \ell_0 + \cdots + \ell_k$. Note for $\ell = \ell_0 + \ell_1 + \cdots \in \mathcal{L}$, and $m = m_k + m_{k+1} + \cdots$, $k \geq 1$, that

$$\exp(\mathrm{ad}(m))(\ell) = J_{k-1}(\ell) + (\ell_k - [\ell_0, m_k]) + \mathcal{O}_{k+1}$$

where $\mathcal{O}_{k+1}$ denotes terms of order greater or equal to $k+1$.

For $k \geq 0$ and $\ell \in \mathcal{L}$ let

$$C_k(\ell) := \{m = m_1 + m_2 + \cdots | \mathrm{ad}(\ell)(m) = J_k(\ell) + \mathcal{O}_{k+1}\}$$

(the k-th order centralizer of ℓ). We have $m \in C_k(\ell)$ iff $\exp(\mathrm{ad}(m))$ fixes $J_k(\ell)$. Now $m \in C_k(\ell)$ iff

$$\sum_{j=1}^{i} [\ell_{i-j}, m_j] = 0, \quad \text{for } i = 1, \cdots, k, \tag{1}$$

Now suppose that we have obtained a normal form up to the order k, *i.e.* $J_k(\ell)$ is in a normal form. We look for a normal form up to order $k+1$. Consider the linear mapping

$$\phi_{k+1}(m) = [\ell_0, m_{k+1}] + [\ell_1, m_k] + \cdots + [\ell_k, m_1],$$

where $m \in C_k$. We have $\phi_{k+1}(m) \in \mathcal{L}_{k+1}$. Assume that $\mathcal{C}_{k+1}$ is a complementary subspace of its image $\mathcal{V}_{k+1}$ in $\mathcal{L}_{k+1}$, so that the following splitting holds

$$\mathcal{L}_{k+1} = \mathcal{V}_{k+1} \oplus \mathcal{C}_{k+1}.$$

Then one can find $m \in C_k(\ell)$ such that $J_{k+1}(\exp(\mathrm{ad}(m))\ell) \in \mathcal{C}_{k+1}$, which is a normal form of order $k+1$.

Remark 1.1. Once a $\mathcal{C}_{k+1}$ is chosen with a given basis, the homogeneous part of degree $k+1$ can be written under the given basis. So the chosen bases of the corresponding $\mathcal{C}_k$ determine a structure or a model of the normal form. A normal form with a determined model is unique. We shall compute the structure of unique normal forms for nilpotent vector fields in dimensions 3 and 4 up to certain orders.

2 Unique normal form with respect to the classical degree in $\mathbf{R}^3$

Let us first consider the classical graded Lie algebra H of vector fields in $\mathbf{R}^3$. The Lie bracket is defined by $[\cdot,\cdot] : H \times H \to H$ where

$$[U, V] = DV \cdot U - DU \cdot V$$

for any $U, V \in H$. For any $k \geq 0$ we denote by H_k the vector space of homogeneous vector fields in $\mathbf{R}^3$ of order $k+1$ with respect to the classical order (total degree). (One remarks that the subscript is k). Then $\{H, [\cdot,\cdot]\}$ forms a graded Lie algebra with respect to the classical order, *ie* $[H_i, H_j] \subset H_{i+j}$.

Let X be a vector field in H with a nilpotent linear part $X_0 = y\partial_x + z\partial_y \in H_0$. We shall study the unique normal form of X in H. In [5], we have proved that, via a near

identity transformation and a linear transformation, one can reduce generically the second order terms of X to the following normal form

$$X_1 = (x^2 + \mu_{2,2}xy + \mu_{2,3}xz)\partial_z$$

where $\mu_{2,2}, \mu_{2,3}$ are parameters. It can be achieved by considering the linear map $\phi_1(m_1) = [X_0, m_1]$ and via a linear change of coordinates. We shall suppose that this has been done and study the normal form of higher order.

We now consider terms of order 3. We need to study the linear map

$$\phi_2(m_1, m_2) = [X_0, m_2] + [X_1, m_1]$$

subject to the condition $[X_0, m_1] = 0$. From $[X_0, m_1] = 0$ one obtains

$$m_1 = \Big(\alpha_1 z^2 + \alpha_2(y^2 - 2xz) + \alpha_3 yz + \alpha_4 xz\Big)\partial_x + (\alpha_3 z^2 + \alpha_4 yz)\partial_y + \alpha_4 z^2 \partial_z.$$

By computing the kernel of ϕ_2, one finds that its dimension is 7 and if $m = m_1 + m_2 \in \ker(\phi_2)$ then $m_1 = \alpha_1 z^2 \partial_x$. Then the image $\mathcal{V}_2$ of ϕ_2 in H_2 is of dimension 27 and hence $\dim \mathcal{C}_2 = 30 - 27 = 3$. One can prove that $\{x^3, x^2y, x^2z\}$ form a basis of a complementary subspace $\mathcal{C}_2$ to the image $\mathcal{V}_2$ of ϕ_2 in H_2. The homogeneous part of degree 3 in a unique normal form can be chosen to be in the form

$$X_2 = \mu_{3,1}x^3 + \mu_{3,2}x^2y + \mu_{3,3}x^2z.$$

The method can be used to compute a basis of $\mathcal{C}_k$ successively for any $k = 2, 3, \cdots$. We have implemented the method in Maple and obtained a basis for all $k \leq 12$ under some non degeneracy conditions (algebraic conditions on the coefficients of the vector field). We state the results in the following proposition.

Proposition 2.1. *Let X be a nilpotent vector field in $\mathbf{R}^3$ as above. Then a non degenerate unique normal form of X up to order 12 is*

$$\begin{aligned}
&\dot{x} = y, \quad \dot{y} = z,\\
&\dot{z} = x^2 + \mu_{2,2}xy + \mu_{2,3}xz + \mu_{3,1}x^3 + \mu_{3,2}x^2y + \mu_{3,3}x^2z + \mu_{4,1}x^4 + \mu_{4,2}x^3y\\
&\quad +\mu_{5,1}x^5 + \mu_{5,2}x^4y + \mu_{5,3}x^4z + \mu_{6,1}x^6 + \mu_{6,2}x^5y + \mu_{6,3}x^5z + \mu_{6,4}x^4y^2\\
&\quad +\mu_{7,1}x^7 + \mu_{7,2}x^6y + \mu_{7,3}x^6z + \mu_{7,4}x^5y^2 + \mu_{7,5}x^5yz + \mu_{8,1}x^8 + \mu_{8,2}x^7y\\
&\quad +\mu_{8,3}x^7z + \mu_{8,4}x^6y^2 + \mu_{9,1}x^9 + \mu_{9,2}x^8y + \mu_{9,3}x^8z + \mu_{9,4}x^7y^2 + \mu_{9,5}x^7yz\\
&\quad +\mu_{9,6}x^7z^2 + \mu_{10,1}x^{10} + \mu_{10,2}x^9y + \mu_{10,3}x^9z + \mu_{10,4}x^8y^2 + \mu_{10,5}x^8yz\\
&\quad +\mu_{11,1}x^{11} + \mu_{11,2}x^{10}y + \mu_{11,3}x^{10}z + \mu_{11,4}x^9y^2 + \mu_{11,5}x^9yz + \mu_{11,6}x^9z^2\\
&\quad +\mu_{12,1}x^{12} + \mu_{12,2}x^{11}y + \mu_{12,3}x^{11}z + \mu_{12,4}x^{10}y^2 + \mu_{12,5}x^{10}yz\\
&\quad +\mu_{12,6}x^{10}z^2 + \mu_{12,7}x^9y^2z + \mathcal{O}_{13}
\end{aligned}$$

In Table 1, we shall show the number of parameters in each degree remaining in a normal form and compare with those obtained by other methods.

3 New ordering of vector fields in $\mathbf{R}^3$

3.1 Linear grading function

The idea of linear grading function is to define a new grading (new ordering) so that H remains a graded Lie algebra with respect to the new ordering. It is first introduced in [10]. For details and properties of linear grading functions we refer to the original paper. The method has been applied to nilpotent vector fields of dimension 2 in [6, 10, 14].

For our purpose we need the following linear grading function: for any $i, j, k \in \mathbf{N}$,

$$\begin{aligned}\delta(x^i y^j z^k \partial_x) &= 3i + 4j + 5k - 3,\\ \delta(x^i y^j z^k \partial_y) &= 3i + 4j + 5k - 4,\\ \delta(x^i y^j z^k \partial_z) &= 3i + 4j + 5k - 5.\end{aligned}$$

Let $\mathcal{H}_k$ ($k \geq 1$) be the vector space spanned by all monomial vector fields of degree k with respect to δ. One has $[\mathcal{H}_s, \mathcal{H}_t] \subset \mathcal{H}_{s+t}$. Hence $\mathcal{H} = \sum_{k\geq 1} \mathcal{H}_k$ is a graded Lie algebra. We have for example

$$\begin{aligned}\mathcal{H}_1 &= \mathrm{Span}\{y\partial_x, z\partial_y, x^2\partial_z\}\\ \mathcal{H}_2 &= \mathrm{Span}\{z\partial_x, x^2\partial_y, xy\partial_z\}\\ \mathcal{H}_3 &= \mathrm{Span}\{x^2\partial_x, xy\partial_y, xz\partial_z, y^2\partial_z\}.\end{aligned}$$

The following lemma can be verified immediately using the definition of δ.

Lemma 3.1. *We have*

$$\begin{aligned}x^i y^j z^k \partial_y \in \mathcal{H}_N \ \text{ iff } \ x^i y^j z^k \partial_x \in \mathcal{H}_{N+1};\\ x^i y^j z^k \partial_z \in \mathcal{H}_N \ \text{ iff } \ x^i y^j z^k \partial_y \in \mathcal{H}_{N+1};\\ x^i y^j z^k \partial_x \in \mathcal{H}_N \ \text{ iff } \ x^{i+1} y^j z^k \partial_z \in \mathcal{H}_{N+1}.\end{aligned}$$

Consider the nilpotent vector field X of the preceding section in $\mathcal{H}$. We can write $X = \sum_{j\geq 1} V_j$, with $V_j \in \mathcal{H}_j$. As in the preceding section one can assume (see [5]) that

$$V_1 = y\partial_x + z\partial_y + x^2\partial_z.$$

Now we need to consider the adjoint map $\mathrm{ad}(V_1) = [V_1, \cdot]$ in the new graded Lie algebra context. We have $[V_1, \cdot] : \mathcal{H}_k \to \mathcal{H}_{k+1}$.

Lemma 3.2. *Let notations be as above. We have* $\ker[V_1, \cdot]|_{\mathcal{H}_1} = \mathrm{Span}\{V_1\}$, *and for any* $2 \leq k \leq 60$, $\ker[V_1, \cdot]|_{\mathcal{H}_k} = \{0\}$, *i.e. the linear map* $[V_1, \cdot]|_{\mathcal{H}_k}$ *is injective.*

Proof. Let $m_1 = \alpha_1 y\partial_x + \alpha_2 z\partial_y + \alpha_3 x^2\partial_z \in \mathcal{H}_1$. One has

$$[V_1, m_1] = (\alpha_1 - \alpha_2)z\partial_x + (\alpha_2 - \alpha_3)x^2\partial_y + 2(\alpha_3 - \alpha_1)xy\partial_z.$$

Then $m_1 \in \ker[V_1, \cdot]|_{\mathcal{H}_1}$ implies $m_1 = \alpha V_1$. Hence $\ker[V_1, \cdot]|_{\mathcal{H}_1} = \mathrm{Span}\{V_1\}$.

This method can be used to compute the kernel of $[V_1, \cdot]$ in any $\mathcal{H}_k$. We have implemented the method in Maple and the computations show that $[V_1, \cdot]$ is injective for any $2 \leq k \leq 60$. □

Now we have $\dim \mathrm{Im}[V_1, \cdot]|_{\mathcal{H}_1} = 2$. If $\mathcal{C}_2$ is a complementary subspace of $\mathrm{Im}[V_1, \cdot]|_{\mathcal{H}_1}$ in $\mathcal{H}_2$, then $\dim \mathcal{C}_2 = 1$. We can prove that $\mathcal{C}_2 = \mathrm{Span}\{xy\partial_z\}$ is such a one. We obtain a normal form in $\mathcal{H}$ whose homogeneous part of degree 2 with respect to δ is $V_2 = \mu_2 xy\partial_z$.

Proposition 3.3. *(a) Let notations be as above. Let $\mathcal{C}_j$ be a complementary subspace to the image of $\mathrm{ad}(V_1)|_{\mathcal{H}_{j-1}}$ in $\mathcal{H}_j$. Then one can reduce the vector field to a normal form $V = V_1 + \sum_{j\geq 2} V_j$ with $V_j \in \mathcal{C}_j$ for $j \geq 2$. Furthermore the normal form is unique up to the order 60.*

(b) One can choose the complementary subspaces to be

$$\mathcal{C}_3 = \mathrm{Span}\{xz\partial_z\},\ \mathcal{C}_4 = \mathrm{Span}\{x^3\partial_z\},\ \mathcal{C}_5 = \mathrm{Span}\{z^2\partial_z\},\ \mathcal{C}_6 = \{0\},$$

and for $7 \leq N \leq 60$,

$$\mathcal{C}_N = \mathrm{Span}\{y^j z^k \partial_z : 4j + 5k - 5 = N\}.$$

Proof. For part (a) it suffices to prove the uniqueness. Let $V = V_1 + \cdots + V_{k-1} + \cdots$ be a normal form up to the order $k-1$ (with $k <= 60$). We now look for a normal form of order k. Let $m = m_1 + m_2 + \cdots + m_k + \cdots$ be such that

$$\begin{aligned}
&[V_1, m_1] = 0,\\
&[V_1, m_2] + [V_2, m_1] = 0,\\
&\cdots\cdots\\
&[V_1, m_{k-1}] + [V_2, m_{k-2}] + \cdots + [V_{k-1}, m_1] = 0.
\end{aligned}$$

Then according to Lemma 3.2, one has $m_1 = \alpha V_1$. We obtain from the other equations that $m_j = \alpha V_j$. Hence

$$\begin{aligned}
&[V_1, m_k] + [V_2, m_{k-1}] + \cdots + [V_{k-1}, m_2] + [V_k, m_1]\\
&\quad = [V_1, m_k - \alpha V_k] + [V_2, \alpha V_{k-1}] + \cdots + [V_{k-1}, \alpha V_2]\\
&\quad = [V_1, m_k - \alpha V_k].
\end{aligned}$$

Finally a normal form obtained by using $\mathrm{ad}(V_1)$ is unique up to the order 60. Notice that in the terminology of [10], our result shows that the *first order* normal form is unique up to the given order.

We have implemented the algorithm in Maple. We have been able to obtain a basis of $\mathcal{C}_N$ for $N \leq 60$ as stated in part (b) of the proposition.

□

We state the corresponding normal form in the following proposition.

Proposition 3.4. *Let X be a nilpotent vector field as above. Then X can be generically reduced to a unique normal form up to order 60 in the new ordering*

$$\begin{aligned}
&\dot{x} = y, \quad \dot{y} = z,\\
&\dot{z} = x^2 + \mu_2 xy + \mu_3 xz + \mu_4 x^3 + \mu_5 z^2 + \sum_{7\leq 4j+5k-5\leq 60} \mu_{j,k} y^j z^k + \mathcal{O}_{61}.
\end{aligned}$$

where $\mathcal{O}_{61}$ represents terms of degree ≥ 61 in $\mathcal{H}$.

The dimension of $\mathcal{C}_N$ is given in the following lemma. This is useful for calculating the number of parameters remaining in the normal form of the above proposition.

Lemma 3.5. *For any $N \geq 2$ let $d = \#\{(j,k) \in \mathbf{N}^2 | 4j+5k-5 = N\}$. Let $N+5 = 20p+s$ with $0 \leq s < 20$. Then for $s = 1,2,3,6,7,11$ we have $d = p$, and for other values of s, we have $d = p+1$.*

Proof. The assertion follows from a calculation of the number of solutions of $4j+5k-5 = N$. For example with $s = 5$ we have, with $j' = j+k$,

$$\begin{aligned} d &= \sum_{4j+5k=20p+5} 1 = \sum_{4j'+k=20p+5} 1 = \sum_{\substack{k=4(5p-j'+1)+1 \\ j=5(j'-4p-1)}} 1 \\ &= \sum_{4p+1 \leq j' \leq 5p+1} 1 = p+1. \end{aligned}$$

The proof is similar for other s. □

In Table 1, the number of parameters in each degree remaining in a normal form in $\mathcal{H}$ is compared with that of Proposition 2.1. It leads to the important conclusion that with this choice of complementary subspace the normal form obtained is not unique with respect to the classical degree. One remarks that the third equation in the normal form of Proposition 3.4 contains all terms of the form $y^j z^k$ with $j+k \geq 3$.

In the following paragraphe we shall show that we can choose another complementary subspace so that the normal form contains the same number of parameters as in the case of the classical degrees.

Total degree	Prop. 5.3 in [5]	Prop. 2.1	Prop. 3.6	Prop. 3.4
3	3	3	3	5
4	2	2	2	5
5	3	3	3	6
6		4	4	7
7		5	5	8
8		4	4	9
9		6	6	10
10		5	5	11
11		6	6	12
12		7	7	13
13			8	14
14			8	15
15			9	16

Table 1. Comparison of the numbers of parameters.

3.2 Second choice of a complementary subspace

In the preceding paragraphe, the basis for $\mathcal{C}_N$ do not contain x. To give another choice of complementary subspaces, we have looked for bases so that the powers in x are the maximum possible. We state the result in the following proposition. It is obtained by an implementation of the method in Maple.

Proposition 3.6. *Let $X \in \mathcal{H}$ be a nilpotent vector field as above. Then X can be generically reduced to a unique normal form up to order 56 in the new grading :*

$$
\begin{aligned}
&\dot{x} = y, \quad \dot{y} = z,\\
&\dot{z} = x^2 + \mu_2 xy + \mu_3 xz + \mu_4 x^3 + \mu_5 x^2 y + \mu_7 x^4 + \mu_8 x z^2 + \mu_9 x^3 z + \mu_{10} x^5\\
&\quad + \mu_{11} x^4 y + \mu_{12} x^4 z + \mu_{13} x^6 + \mu_{14} x^5 y + \mu_{15} x^4 y^2 + \mu'_{15} x^5 z + \mu_{16} x^7\\
&\quad + \mu_{17} x^6 y + \mu_{18} x^6 z + \mu_{19} x^5 yz + \mu'_{19} x^8 + \mu_{20} x^5 z^2 + \mu'_{20} x^7 y + \mu_{21} x^7 z\\
&\quad + \mu_{22} x^9 + \mu_{23} x^6 z^2 + \mu'_{23} x^8 y + \mu_{24} x^7 y^2 + \mu'_{24} x^8 z + \mu_{25} x^7 yz + \mu'_{25} x^{10}\\
&\quad + \mu_{26} x^9 y + \mu_{27} x^8 y^2 + \mu'_{27} x^9 z + \mu_{28} x^6 z^3 + \mu'_{28} x^{11} + \mu_{29} x^8 z^2 + \mu'_{29} x^{10} y\\
&\quad + \mu_{30} x^9 y^2 + \mu'_{30} x^{10} z + \mu_{31} x^9 yz + \mu'_{31} x^{12} + \mu_{32} x^9 z^2 + \mu'_{32} x^{11} y + \mu_{33} x^{10} y^2\\
&\quad + \mu'_{33} x^{11} z + \mu_{34} x^{10} yz + \mu'_{34} x^{13} + \mu_{35} x^9 y^2 z + \mu'_{35} x^{10} z^2 + \mu''_{35} x^{12} y\\
&\quad + \mu_{36} x^{11} y^2 + \mu'_{36} x^{12} z + \mu_{37} x^{11} yz + \mu'_{37} x^{14} + \mu_{38} x^{11} z^2 + \mu'_{38} x^{13} y\\
&\quad + \mu_{39} x^{10} yz^2 + \mu'_{39} x^{12} y^2 + \mu''_{39} x^{13} z + \mu_{40} x^{10} z^3 + \mu'_{40} x^{12} yz + \mu''_{40} x^{15}\\
&\quad + \mu_{41} x^{12} z^2 + \mu'_{41} x^{14} y + \mu_{42} x^{13} y^2 + \mu'_{42} x^{14} z + \mu_{43} x^{11} z^3 + \mu'_{43} x^{13} yz\\
&\quad + \mu''_{43} x^{16} + \mu_{44} x^{12} y^2 z + \mu'_{44} x^{13} z^2 + \mu''_{44} x^{15} y + \mu_{45} x^{12} yz^2 + \mu'_{45} x^{14} y^2\\
&\quad + \mu''_{45} x^{15} z + \mu_{46} x^{14} yz + \mu'_{46} x^{17} + \mu_{47} x^{13} y^2 z + \mu'_{47} x^{14} z^2 + \mu''_{47} x^{16} y\\
&\quad + \mu_{48} x^{11} z^4 + \mu'_{48} x^{15} y^2 + \mu''_{48} x^{16} z + \mu_{49} x^{13} z^3 + \mu'_{49} x^{15} yz + \mu''_{49} x^{18}\\
&\quad + \mu_{50} x^{14} y^2 z + \mu'_{50} x^{15} z^2 + \mu''_{50} x^{17} y + \mu_{51} x^{14} yz^2 + \mu'_{51} x^{16} y^2 + \mu''_{51} x^{17} z\\
&\quad + \mu_{52} x^{14} z^3 + \mu'_{52} x^{16} yz + \mu''_{52} x^{19} + \mu_{53} x^{15} y^2 z + \mu_{53} x^{16} z^2 + \mu_{53} x^{18} y\\
&\quad + \mu_{54} x^{15} yz^2 + \mu'_{54} x^{17} y^2 + \mu''_{54} x^{18} z + \mu_{55} x^{14} y^2 z^2 + \mu'_{55} x^{15} z^3 + \mu''_{55} x^{17} yz\\
&\quad + \mu'''_{55} x^{20} + \mu_{56} x^{16} y^2 z + \mu'_{56} x^{17} z^2 + \mu''_{56} x^{19} y + \mathcal{O}_{57}
\end{aligned}
$$

Remark 3.7. In Table 1, the numbers of parameters in each degree remaining in the normal forms of Propositions 2.1, 3.4 and 3.6 are compared for certain degrees.

One can conclude that a unique normal form obtained with respect to a new ordering may not be a unique normal form with respect to the classical order. This gives a negative answer to the question raised in the introduction.

Remark 3.8. According to Lemma 3.2, one may conjecture that the adjoint map $[V_1, \cdot]$ is injective in $\mathcal{H}_k$ for any $k \geq 2$. This is an interesting phenomena since it is not true in dimensions 2 and 4 with respect to the linear grading function (see the following section for the case of dimension 4). In fact in dimension 2 (see [6]) we have defined a new grading as follows, for any $i, j \in \mathbf{N}$,

$$
\begin{aligned}
\delta'(x^i y^j \partial_x) &= 2i + 3j - 2,\\
\delta'(x^i y^j \partial_y) &= 2i + 3j - 3.
\end{aligned}
$$

One then has generically $V'_1 = y\partial_x + x^2 \partial_y$. We have proved (see [6]) that $[V'_1, \cdot]|_{\mathcal{H}_k}$ is not injective for any $k = 6p + 1$, in which case

$$\ker[V_1', \cdot]|_{\mathcal{H}_{6p+1}} = \mathrm{Span}\{(\frac{x^3}{3} - \frac{y^2}{2})^p V_1'\}.$$

4 Nilpotent vector fields in dimension 4 with respect to a new ordering

Now we consider vector fields in dimension 4. We define a similar new ordering $\tilde{\delta}$ as follows:

$$\begin{aligned}
\tilde{\delta}(x_1^i x_2^j x_3^k x_4^r \partial_1) &= 4i + 5j + 6k + 7r - 4,\\
\tilde{\delta}(x_1^i x_2^j x_3^k x_4^r \partial_2) &= 4i + 5j + 6k + 7r - 5,\\
\tilde{\delta}(x_1^i x_2^j x_3^k x_4^r \partial_3) &= 4i + 5j + 6k + 7r - 6,\\
\tilde{\delta}(x_1^i x_2^j x_3^k x_4^r \partial_4) &= 4i + 5j + 6k + 7r - 7,
\end{aligned}$$

where $\partial_i = \partial/\partial x_i$ for $1 \leq i \leq 4$. We denote by $\tilde{\mathcal{H}}_N$ the space of vector fields of homogeneous degree N with respect to the new ordering. Similarly to the three dimensional case, one has generically $\tilde{V}_1 = x_2\partial_1 + x_3\partial_2 + x_4\partial_3 + x_1^2\partial_4 \in \tilde{\mathcal{H}}_1$. We now consider the adjoint linear map $[\tilde{V}_1, \cdot]|_{\tilde{\mathcal{H}}_j}$. By computations in Maple we have proved that it is not injective for $j = 13, 25$ and 37. Therefore Lemma 3.2 shows that there is an interesting phenomena for nilpotent vector fields in dimension 3 which is different from the dimensional 2 and 4 cases.

In fact we have $\ker[\tilde{V}_1, \cdot]|_{\tilde{\mathcal{H}}_1} = \mathrm{Span}\{\tilde{V}_1\}$ similarly to the three dimensional case. One can compute a unique normal form up to the order 13 by using the linear map $[\tilde{V}_1, \cdot]$. For example we obtain a normal form up to order 2 with $\tilde{V}_2 = \mu_2 x_1 x_2 \partial_4$. We have $\dim\ker[\tilde{V}_1, \cdot]|_{\tilde{\mathcal{H}}_{13}} = 1$ with

$$\begin{aligned}
\ker[\tilde{V}_1, \cdot]|_{\tilde{\mathcal{H}}_{13}} = \mathrm{Span}\Big\{&(\frac{3}{2}x_2x_3^2 - 3x_2^2x_4 + x_1^3x_2)\partial_1 + (\frac{3}{2}x_3^3 - 3x_2x_3x_4 + x_1^3x_3)\partial_2\\
&+(-3x_2x_4^2 + \frac{3}{2}x_3^2x_4 + x_1^3x_4)\partial_3 + (-3x_1^2x_2x_4 + \frac{3}{2}x_1^2x_3^2 + x_1^5)\partial_4\Big\}.
\end{aligned}$$

Therefore the image of $[\tilde{V}_1, \cdot]|_{\tilde{\mathcal{H}}_{13}}$ is of dimension $\dim\tilde{\mathcal{H}}_{13} - 1 = 19$ and the dimension of its complementary subspace is $\dim\tilde{\mathcal{H}}_{14} - 19 = 3$. We then compute a basis of a complementary subspace to be $\{x_1^4x_2\partial_4, x_1^2x_3x_4\partial_4, x_1x_2x_3^2\partial_4\}$.

To find a unique normal form of order 15 we need to consider the following equations

$$[\tilde{V}_1, m_{13}] = 0, \quad [\tilde{V}_1, m_{14}] + [\tilde{V}_2, m_{13}] = 0.$$

Under some non degeneracy condition (here $\mu_2 \neq 0$) one has $m_{13} = 0$ and $m_{14} = 0$. Hence the dimension of the image of the linear map $\phi_2(m_{13}, m_{14})$ in $\tilde{\mathcal{H}}_{15}$ is $\dim\tilde{\mathcal{H}}_{14} + 1 = 23$ and the dimension of its complementary subspace is $\dim\tilde{\mathcal{H}}_{15} - 23 = 1$. One computes then a complementary subspace to be $\mathrm{Span}\{x_1^4x_3\partial_4\}$.

Similar arguments can be done to obtain a unique normal form for the order 27 and 39. We have implemented the algorithm in Maple and obtained a unique normal form in $\tilde{\mathcal{H}}$ up to the order 40 which we state in the following.

Proposition 4.1. *Consider a vector field X in dimension 4 with a nilpotent linear part of index 1, i.e. $X_1 = x_2\partial_1 + x_3\partial_2 + x_4\partial_3$. Then a non degenerate unique normal form of X up to the order 40 with respect to the new grading $\tilde{\delta}$ is*

$$
\begin{aligned}
&\dot{x}_1 = x_2, \qquad \dot{x}_2 = x_3, \qquad \dot{x}_3 = x_4,\\
&\dot{x}_4 = \mu_1 x_1^2 + \mu_2 x_1x_2 + \mu_3 x_1x_3 + \mu_4 x_1x_4 + \mu_5 x_1^3 + \mu_5^{(2)} x_3^2 + \mu_6 x_1^2x_2\\
&\quad +\mu_7 x_1^2x_3 + \mu_8 x_1^2x_4 + \mu_9 x_1^4 + \mu_{10} x_1^3x_2 + \mu_{10}^{(2)} x_1x_3x_4 + \mu_{11} x_1^3x_3\\
&\quad +\mu_{11}^{(2)} x_1^2x_2^2 + \mu_{12} x_1^3x_4 + \mu_{12}^{(2)} x_1^2x_2x_3 + \mu_{13} x_1^5 + \mu_{13}^{(2)} x_1^2x_3^2 + \mu_{14} x_1^4x_2\\
&\quad +\mu_{14}^{(2)} x_1^2x_3x_4 + \mu_{14}^{(3)} x_1x_2x_3^2 + \mu_{15} x_1^4x_3 + \mu_{16} x_1^4x_4 + \mu_{16}^{(2)} x_1^3x_2x_3 + \mu_{17} x_1^6\\
&\quad +\mu_{17}^{(2)} x_1^3x_3^2 + \mu_{17}^{(3)} x_1^3x_2x_4 + \mu_{18} x_1^5x_2 + \mu_{18}^{(2)} x_1^3x_3x_4 + \mu_{18}^{(3)} x_1^2x_2x_3^2 + \mu_{19} x_1^5x_3\\
&\quad +\mu_{19}^{(2)} x_1^4x_2^2 + \mu_{19}^{(3)} x_1^3x_4^2 + \mu_{20} x_1^5x_4 + \mu_{20}^{(2)} x_1^4x_2x_3 + \mu_{20}^{(3)} x_1^2x_3^2x_4 + \mu_{21} x_1^7\\
&\quad +\mu_{21}^{(2)} x_1^4x_3^2 + \mu_{21}^{(3)} x_1^4x_2x_4 + \mu_{22} x_1^6x_2 + \mu_{22}^{(2)} x_1^4x_3x_4 + \mu_{22}^{(3)} x_1^3x_2x_3^2 + \mu_{23} x_1^6x_3\\
&\quad +\mu_{23}^{(2)} x_1^5x_2^2 + \mu_{23}^{(3)} x_1^4x_4^2 + \mu_{23}^{(4)} x_1^3x_3^3 + \mu_{24} x_1^6x_4 + \mu_{24}^{(2)} x_1^5x_2x_3 + \mu_{24}^{(3)} x_1^4x_2^3\\
&\quad +\mu_{24}^{(4)} x_1^3x_3^2x_4 + \mu_{25} x_1^8 + \mu_{25}^{(2)} x_1^5x_3^2 + \mu_{25}^{(3)} x_1^5x_2x_4 + \mu_{25}^{(4)} x_1^3x_3x_4^2 + \mu_{26} x_1^7x_2\\
&\quad +\mu_{26}^{(2)} x_1^5x_3x_4 + \mu_{26}^{(3)} x_1^4x_2x_3^2 + \mu_{26}^{(4)} x_1^4x_2^2x_4 + \mu_{26}^{(5)} x_1^3x_2^3x_3 + \mu_{27} x_1^7x_3\\
&\quad +\mu_{27}^{(2)} x_1^6x_2^2 + \mu_{27}^{(3)} x_1^5x_4^2 + \mu_{28} x_1^7x_4 + \mu_{28}^{(2)} x_1^6x_2x_3 + \mu_{28}^{(3)} x_1^5x_2^3 + \mu_{28}^{(4)} x_1^4x_3^2x_4\\
&\quad +\mu_{28}^{(5)} x_1^4x_2x_4^2 + \mu_{29} x_1^9 + \mu_{29}^{(2)} x_1^6x_3^2 + \mu_{29}^{(3)} x_1^6x_2x_4 + \mu_{29}^{(4)} x_1^5x_2^2x_3 + \mu_{29}^{(5)} x_1^4x_3x_4^2\\
&\quad +\mu_{30} x_1^8x_2 + \mu_{30}^{(2)} x_1^6x_3x_4 + \mu_{30}^{(3)} x_1^5x_2x_3^2 + \mu_{30}^{(4)} x_1^5x_2^2x_4 + \mu_{30}^{(5)} x_1^4x_4^3 + \mu_{31} x_1^8x_3\\
&\quad +\mu_{31}^{(2)} x_1^7x_2^2 + \mu_{31}^{(3)} x_1^6x_4^2 + \mu_{31}^{(4)} x_1^5x_3^3 + \mu_{31}^{(5)} x_1^5x_2x_3x_4 + \mu_{32} x_1^8x_4 + \mu_{32}^{(2)} x_1^7x_2x_3\\
&\quad +\mu_{32}^{(3)} x_1^5x_3^2x_4 + \mu_{32}^{(4)} x_1^5x_2x_4^2 + \mu_{32}^{(5)} x_1^4x_2x_3^3 + \mu_{33} x_1^{10} + \mu_{33}^{(2)} x_1^7x_3^2 + \mu_{33}^{(3)} x_1^7x_2x_4\\
&\quad +\mu_{33}^{(4)} x_1^6x_2^2x_3 + \mu_{33}^{(5)} x_1^5x_3x_4^2 + \mu_{33}^{(6)} x_1^4x_3^4 + \mu_{34} x_1^9x_2 + \mu_{34}^{(2)} x_1^7x_3x_4\\
&\quad +\mu_{34}^{(3)} x_1^6x_2x_3^2 + \mu_{34}^{(4)} x_1^6x_2^2x_4 + \mu_{34}^{(5)} x_1^5x_2^3x_3 + \mu_{34}^{(6)} x_1^5x_4^3 + \mu_{35} x_1^9x_3 + \mu_{35}^{(2)} x_1^8x_2^2\\
&\quad +\mu_{35}^{(3)} x_1^7x_4^2 + \mu_{35}^{(4)} x_1^6x_3^3 + \mu_{35}^{(5)} x_1^6x_2x_3x_4 + \mu_{35}^{(6)} x_1^5x_2^2x_3^2 + \mu_{35}^{(7)} x_1^5x_2^3x_4\\
&\quad +\mu_{36} x_1^9x_4 + \mu_{36}^{(2)} x_1^8x_2x_3 + \mu_{36}^{(3)} x_1^7x_2^3 + \mu_{36}^{(4)} x_1^6x_3^2x_4 + \mu_{36}^{(5)} x_1^6x_2x_4^2\\
&\quad +\mu_{36}^{(6)} x_1^5x_2x_3^3 + \mu_{37} x_1^{11} + \mu_{37}^{(2)} x_1^8x_3^2 + \mu_{37}^{(3)} x_1^8x_2x_4 + \mu_{37}^{(4)} x_1^7x_2^2x_3 + \mu_{37}^{(5)} x_1^6x_3x_4^2\\
&\quad +\mu_{37}^{(6)} x_1^5x_3^4 + \mu_{38} x_1^{10}x_2 + \mu_{38}^{(2)} x_1^8x_3x_4 + \mu_{38}^{(3)} x_1^7x_2x_3^2 + \mu_{38}^{(4)} x_1^7x_2^2x_4\\
&\quad +\mu_{38}^{(5)} x_1^6x_2^3x_3 + \mu_{38}^{(6)} x_1^6x_4^3 + \mu_{38}^{(7)} x_1^5x_3^3x_4 + \mu_{38}^{(8)} x_1^5x_2x_3x_4^2 + \mu_{39} x_1^{10}x_3\\
&\quad +\mu_{39}^{(2)} x_1^9x_2^2 + \mu_{39}^{(3)} x_1^8x_4^2 + \mu_{39}^{(4)} x_1^7x_3^3 + \mu_{39}^{(5)} x_1^7x_2x_3x_4 + \mu_{39}^{(6)} x_1^6x_2^2x_3^2\\
&\quad +\mu_{40} x_1^{10}x_4 + \mu_{40}^{(2)} x_1^9x_2x_3 + \mu_{40}^{(3)} x_1^8x_2^3 + \mu_{40}^{(4)} x_1^7x_3^2x_4 + \mu_{40}^{(5)} x_1^7x_2x_4^2\\
&\quad +\mu_{40}^{(6)} x_1^6x_2x_3^3 + \mu_{40}^{(7)} x_1^6x_2^2x_3x_4 + \mu_{40}^{(8)} x_1^5x_3x_4^3 + \mathcal{O}_{41}
\end{aligned}
$$

In [5], a unique normal form with respect to the classical degree up to the order 3 is given (see Proposition 5.5 in [5]). The homogeneous part of both degree 2 and 3 has 5 terms which are the same as in the above proposition.

References

[1] V. I. Arnold, *Geometrical methods in the theory of ordinary differential equations*, New York, Springer-Verlag, 1983.

[2] A. Baider, Unique normal forms for vector fields and Hamiltonians, *J. Diff. Equa.* **78** (1989), 33-52.

[3] A. Baider and J. A. Sanders, Further reductions of the Takens-Bogdanov normal form, *J. Diff. Equa.* **99** (1992), 205-244.

[4] G. Chen, Further reduction of normal forms for vector fields, *Numer. Algo.*, **27** (2001), 1-33.

[5] G. Chen and J. Della Dora, An algorithm for computing a new normal form for dynamical systems, *J. Symb. Comp.*, **29**(3) (2000), 393-418.

[6] G. Chen, D. Wang and X. Wang, Unique normal forms for nilpotent planar vector fields, *Inter. J. Bifur. Chaos*, **12** (2002), 2159-2174.

[7] R. C. Churchill and M. Kummer, A unified approach to linear and nonlinear normal forms for Hamiltonian systems, *J. Symb. Comp.* **27** (1999), 49-131.

[8] H. Dulac, Solution d'un système d'équations différentielles dans le voisinage des valeurs singulières, *Bull. Soc. Math. France*, **40** (1912), 324-383.

[9] G. Gaeta, Reduction of Poincaré normal forms, *Lett. Math. Phys.* **42** (1997), 103-114.

[10] H. Kokubu, H, Oka and D. Wang, Linear grading function and further reduction of normal forms, *J. Diff. Equa.* **132** (1996), 293-318.

[11] H. Poincaré, Notes sur les propriétés des fonctions définies par des équations différentielles, *Journal de l'Ecole Polytechnique*, **45**^{e} cahier, 1878, 13-26.

[12] F. Takens, Singularities of Vector Fields, *Publ. Math., IHES* **43** (1974), 47-100.

[13] S. Ushiki, Normal forms for singularities of vector fields, *Japan J. Appl. Math.* **1** (1984), 1-34.

[14] D. Wang, J. Li, M. Huang and Y. Jiang, Unique normal forms of Bogdanov-Takens singularities, *J. Diff. Equa.*, **163** (2000), 223-238.

[15] J. Yang, Polynomial normal forms for vector fields on $\mathbf{R}^3$, *Duke Math. J.*, **106** (2000), 1-18.

[16] P. Yu and Y. Yuan, The simplest normal form associated with a triple zero eigenvalue of indices one and two, *Nonlinear Anal.* 47 (2001), 1105-1116.

Guoting CHEN
Laboratoire P. Painlevé, UMR CNRS 8524,
UFR de Mathématiques, Université de Lille 1,
59655 Villeneuve d'Ascq, France
gchen@math.univ-lille1.fr

A New Operation on Words Suggested by DNA Biochemistry: Hairpin Completion

Daniela Cheptea Carlos Martín-Vide Victor Mitrana

Abstract

In this paper we propose a new formal operation on words and languages suggested by DNA manipulation, called *hairpin completion*. By this operation, based on a Watson-Crick-like complementarity, one can generate a set of words, starting from a single word. Specifically, starting from one single stranded molecule x such that either a suffix or a prefix of x is complementary to a subword of x a new word z, which is a prolongation of x to the right or to the left, respectively, is obtained by annealing. This operation is considered here as an abstract operation on formal languages. We settle the closure properties under the non-iterated version of this operation of some classes in the Chomsky hierarchy as well as some complexity classes. Then the effect of iterated version on the space complexity classes is investigated. As an immediate consequence, one gets that the iterated hairpin completion of every regular language is polynomially recognizable.

1 Introduction

A DNA molecule consists of a double strand, each DNA single strand being composed by nucleotides which differ from each other by their bases: A (adenine), G (guanine), C (cytosine), and T (thymine). The two strands which form the DNA molecule are kept together by the hydrogen bond between the bases: A always bonds with T, while C with G. This paradigm of *Watson-Crick complementarity* will be one of the main concepts used in defining the formal operation of hairpin completion investigated in the present paper.

Two other biological principles used as sources of inspiration in this paper are that of *annealing* and that of *lengthening DNA by polymerases*. The first principle refers to fusing two single stranded molecules by complementary base pairing while the second one refers to adding nucleotides to one strand (in a more general setting to both strands) of a double stranded DNA molecule. The former operation requires a heated solution containing the two strands which is cooled down slowly. The latter one requires two single strands such that one (usually called *primer*) is bonded to a part of the other (usually called *template*) by Watson-Crick complementarity and a *polymerization buffer* with many copies of the four nucleotides that polymerases will concatenate to the primer by complementing the template.

On the other hand, it is known that a single stranded DNA molecule might produce a hairpin structure. In many DNA-based algorithms, these DNA molecules cannot be used in the subsequent computations. Hairpin or hairpin-free DNA structures have numerous

applications to DNA computing and molecular genetics. In a series of papers (see, e.g., [4, 6, 7]) the problem of finding sets of DNA sequences which are unlikely to lead to "bad" hybridizations is considered. On the other hand, these molecules which may form a hairpin structure have been used as the basic feature of a new computational model reported in [16], where an instance of the 3-SAT problem has been solved by a DNA-algorithm in which the second phase is mainly based on the elimination of hairpin structured molecules. Different types of hairpin and hairpin-free languages are defined in [14], [3], and more recently in [10], where they are studied from a language theoretical point of view.

We now informally explain the hairpin completion operation and how it can be related to the aforementioned biological concepts. Let us consider the following hypothetical biological situation: we are given one single stranded DNA molecule z such that either a prefix or a suffix of z is Watson-Crick complementary to a subword of z. Then the prefix or suffix of z and the corresponding subword of z get annealed by complementary base pairing and then z is lengthened by DNA polymerases up to a complete hairpin structure. The mathematical expression of this hypothetical situation defines the hairpin completion operation. Assume that we have an alphabet and a complementary relation on its letters. The hairpin completion operation, which is a unary operation, might be defined as follows:

(i) a word $w = \alpha\beta\overline{\alpha}^R\gamma$ in which a hairpin structure determined by the complementarity of α and $\overline{\alpha}^R$ appears produces the new word $\overline{\gamma}^R\alpha\beta\overline{\alpha}^R\gamma$.

(ii) a word $w = \gamma\alpha\beta\overline{\alpha}^R$ in which a hairpin structure determined by the complementarity of α and $\overline{\alpha}^R$ appears produces the new word $\gamma\alpha\beta\overline{\alpha}^R\overline{\gamma}^R$.

This operation is schematically illustrated in Figure 1.

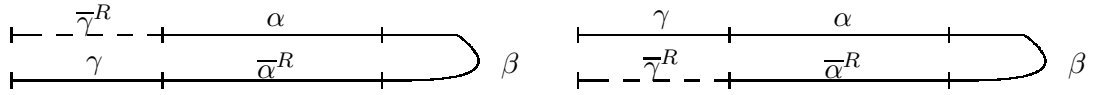


Figure 1: Hairpin completion

Of course, all these phenomena are considered here in an idealized way. For instance, we allow polymerase to extend in either end (3' or 5') despite that, due to the greater stability of 3' when attaching new nucleotides, DNA polymerase can act continuously only in the 5'$\longrightarrow$ 3' direction. However, polymerase can also act in the opposite direction, but in short "spurts" (Okazaki fragments). Moreover, in order to have a "stable" hairpin structure the subword x should be sufficiently long.

This operation is considered here as an abstract operation on formal languages. This line of research lies within a vividly investigated area, that of operations on words and languages suggested and motivated by the biocomputing and bioinformatics fields:

sticking investigated in [9, 5, 12] (a particular type of polyominoes with sticky ends are combined provided that the sticky ends are Watson-Crick complementary),

PA-matching considered in [11] which is related to both the *splicing* (an operation introduced in [8] that opened a broad area of intensive research) and the *annealing* operations,

superposition operation introduced in [1] (two words which may contain *transparent* positions are aligned one over the other and the resulting word is obtained by reading the visible positions as well as aligned transparent positions). A rather different superposition based on a Watson-Crick-like complementarity is proposed in [2], where one generates a set of words, starting from a pair of words, in which the contribution of a word to the result need not be one subword only, as happens in classical bio-operations of DNA computing [13].

The paper is organized as follows: the next section gives the definitions of the basic notions and concepts. Then we consider the non-iterated hairpin completion and show that the hairpin completion of a regular language is not necessarily regular but context-free, while the hairpin completion of a context-free is not necessarily context-free but context-sensitive. Actually, we prove that each space complexity class $NSPACE(f(n))$ is closed under hairpin completion as soon as $f(n) \geq \log n$ is space-constructible. Moreover, both time complexity classes **P** and **NP** are closed under hairpin completion. The last section presents two results concerning iterated hairpin completion. The first one extends the results regarding the behavior of space complexity classes under hairpin completion to the iterated version. As a consequence, we obtain that the iterated hairpin completion of every regular language is polynomially recognizable.

2 Preliminaries

We assume the reader to be familiar with the fundamental concepts of formal language theory and automata theory, particularly the notions of grammar and finite automaton [15].

An alphabet is always a finite set of letters. For a finite set A we denote by $card(A)$ the cardinality of A. The set of all words over an alphabet V is denoted by V^*. The empty word is written ε; moreover, $V^+ = V^* \setminus \{\varepsilon\}$. Given a word w over an alphabet V, we denote by $|w|$ its length, while $|w|_a$ denotes the number of occurrences of the letter a in w. If $w = xyz$ for some $x, y, z \in V^*$, then x, y, z are called prefix, subword, suffix, respectively, of w.

Let Ω be a "superalphabet", that is an infinite set such that any alphabet considered in this paper is a subset of Ω. In other words, Ω is the *universe* of the languages in this paper, i.e., all words and languages are over alphabets that are subsets of Ω. An *involution* over a set S is a bijective mapping $\sigma : S \longrightarrow S$ such that $\sigma = \sigma^{-1}$. Any involution σ on Ω such that $\sigma(a) \neq a$ for all $a \in \Omega$ is said to be here a *Watson-Crick involution*. Despite that this is nothing more than a fixed point-free involution, we prefer this terminology since the hairpin completion defined later is inspired by the DNA lengthening by polymerases, where the Watson-Crick complementarity plays an important role. Let $\bar{\cdot}$ be a Watson-Crick involution fixed for the rest of the paper. The Watson-Crick involution is extended to a morphism from Ω^* to Ω^* in the usual way. We say that the letters a and $\overline{a}$ are complementary to each other. For an alphabet V, we set $\overline{V} = \{\overline{a} \mid a \in V\}$. Note that V and $\overline{V}$ can intersect and they can be, but need not be, equal. Remember that the DNA alphabet consists of

four letters, $V_{DNA} = \{A, C, G, T\}$, which are abbreviations for the four nucleotides and we may set $\overline{A} = T$, $\overline{C} = G$.

We denote by $(\cdot)^R$ the mapping defined by ${}^R : V^* \longrightarrow V^*$, $(a_1 a_2 \dots a_n)^R = a_n \dots a_2 a_1$. Note that ^{R} is an involution and an *anti-morphism* ($(xy)^R = y^R x^R$ for all $x, y \in V^*$). Note also that the two mappings $\bar{\cdot}$ and $\cdot^R$ commutes, namely, for any string x, $(\overline{x})^R = \overline{x^R}$ holds.

A *finite transducer* is a 6-tuple $M = (Q, V_i, V_o, q_0, F, \delta)$ where Q, V_i, V_o are finite and nonempty sets (the set of states, the input alphabet, and the output alphabet, respectively), $q_0 \in Q$ (the initial state), $F \subseteq Q$ (the set of final states), and δ is the (transition-and-output) function from $Q \times (V_i \cup \{\varepsilon\})$ to finite subsets of $Q \times V_o^*$. This function is extended in a natural way to $Q \times V_i^*$. Every finite transducer M as above defines a *finite transduction*

$$M(\alpha) = \{\beta \in V_o^* \mid \text{ there exists } q \in F \text{ such that } (q, \beta) \in \delta(q_0, \alpha)\},\ \alpha \in V_i^*.$$

If $M(\alpha) \neq \emptyset$, then we say that α is "accepted" by M. The language accepted by a finite transducer M is the set of all α such that $M(\alpha) \neq \emptyset$. The finite transduction M is extended to languages $L \subseteq V_i^*$ in the obvious way, namely $M(L) = \bigcup_{\alpha \in L} M(\alpha)$. If we ignore V_o and the output part of δ, then we obtain a *finite automaton* (with ε moves). A finite automaton is denoted (Q, V, q_0, F, δ). A language is *regular* iff it is accepted by a finite automaton.

Definition 2.1. Let V be an alphabet, for any $w \in V^+$ we define the *k-hairpin completion* of w, denoted by $\rightarrow_k$, for some $k \geq 1$, as follows:

$$\begin{aligned} w \rightarrow_k &= \{\overline{\gamma^R} w | w = \alpha\beta\overline{\alpha^R}\gamma, |\alpha| = k,\ \alpha, \beta \in V^+, \gamma \in V^*\} \\ w \rightarrow_k &= \{w\overline{\gamma^R} | w = \gamma\alpha\beta\overline{\alpha^R}, |\alpha| = k,\ \alpha, \beta \in V^+, \gamma \in V^*\} \\ w \rightarrow_k &= w \rightarrow_k \cup w \rightarrow_k \end{aligned}$$

The *hairpin completion* of w is defined by

$$w \rightarrow = \bigcup_{k \geq 1} w \rightarrow_k .$$

Clearly, $w \rightarrow_{k+1} \subseteq w \rightarrow_k$ for any $w \in V^+$ and $k \geq 1$, hence $w \rightarrow = w \rightarrow_1$. The hairpin completion operation is naturally extended to languages by

$$L \rightarrow_k = \bigcup_{w \in L} w \rightarrow_k \qquad L \rightarrow = \bigcup_{w \in L} w \rightarrow .$$

The iterated version of the hairpin completions is defined as usual by:

$$\begin{array}{lll} w(\rightarrow_k)^0 = \{w\}, & w(\rightarrow_k)^{n+1} = (w(\rightarrow_k)^n) \rightarrow_k, & w(\rightarrow_k)^* = \bigcup_{n \geq 0} w(\rightarrow_k)^n \\ w(\rightarrow)^0 = \{w\}, & w(\rightarrow)^{n+1} = (w(\rightarrow)^n) \rightarrow, & w(\rightarrow)^* = \bigcup_{n \geq 0} w(\rightarrow)^n \\ L(\rightarrow_k)^* = \bigcup_{w \in L} w(\rightarrow_k)^* & & L(\rightarrow)^* = \bigcup_{w \in L} w(\rightarrow)^*. \end{array}$$

3 Non-iterated Hairpin Completion

In this section we consider the non-iterated hairpin completion as a formal operation on languages. A family of languages $\mathcal{F}$ is closed under hairpin completion if the hairpin completion of any language from $\mathcal{F}$ lies in $\mathcal{F}$. We show that the class of regular and context-free languages are not closed under hairpin completion but almost all space complexity classes are closed under hairpin completion.

Theorem 3.1. *For any integer $k \geq 1$, a language is linear if and only if it is the morphic image of the k-hairpin completion of a regular language.*

Proof. Let $k \geq 1$ and L be a regular language over the alphabet V. Let further α be a word in V^* of length k. We define the following two languages $L_r(\alpha) = L \cap \{\gamma\alpha\beta\overline{\alpha^R} | \beta \in V^+, \gamma \in V^*\}$ and $L_l(\alpha) = L \cap \{\alpha\beta\overline{\alpha^R}\gamma | \beta \in V^+, \gamma \in V^*\}$ which are obviously regular languages. One can easily observe that

$$L \rightarrow_k = \bigcup_{\alpha \in V^*,\ |\alpha|=k} ((L_r(\alpha) \rightarrow_k) \cup (L_l(\alpha) \rightarrow_k)).$$

We prove that both languages $L_r(\alpha) \rightarrow_k$ and $L_l(\alpha) \rightarrow_k$ are linear for every $\alpha \in V^*$ with $|\alpha| = k$, therefore $L \rightarrow_k$ is still linear as a finite union of linear languages. To this aim, we construct a linear grammar generating $L_r(\alpha) \rightarrow_k$ for an arbitrary α of length k. A similar construction for the language $L_l(\alpha) \rightarrow_k$ is left to the reader.

Let $G = (N, V, S, P)$ be a regular grammar generating $L_r(\alpha)$. We define the linear grammar $G' = (N', V \cup \overline{V}, S, P')$, where:

$$\begin{aligned} N' \;=\; & N \cup \{A' \mid A \in N\} \cup (N \times \{[x] \mid x \text{ is a suffix of } \alpha\}) \cup \\ & (N \times \{\langle x \rangle \mid x \text{ is a suffix of } \overline{\alpha}^R\}), \end{aligned}$$

and the set of rules P' is defined as follows:

- For each rule $A \rightarrow aB \in P$ the set P' contains all rules:

$$\begin{array}{ll} A \rightarrow aB\overline{a} & A \rightarrow a(B, [\alpha])\overline{a} \\ (A, [ax]) \rightarrow a(B, [x]), x \in V^+ & (A, [a]) \rightarrow aB' \\ A' \rightarrow aB' & A' \rightarrow a(B, \langle\overline{\alpha}^R\rangle) \\ (A, \langle ax\rangle) \rightarrow a(B, \langle x\rangle), x \in V^* & \end{array}$$

- For each rule $A \rightarrow a \in P$ the set P' contains the rule $(A, \langle a\rangle) \rightarrow a$.

For any $\gamma\alpha\beta\overline{\alpha^R\gamma^R} \in L_r(\alpha) \rightarrow_k$, the string $\gamma\alpha\beta\overline{\alpha^R} \in L_r(\alpha)$, hence $\gamma\alpha\beta\overline{\alpha^R}$ is generated by G. Note that $\beta \neq \varepsilon$. It follows that $S \Longrightarrow^* \gamma A_1 \Longrightarrow^* \gamma\alpha A_2 \Longrightarrow^* \gamma\alpha\beta A_3 \Longrightarrow^* \gamma\alpha\beta\overline{\alpha^R}$ and the following derivation is possible in G':

$$S \Longrightarrow^* \gamma(A_1, [\alpha])\overline{\gamma}^R \Longrightarrow^* \gamma\alpha A_2'\gamma^R \Longrightarrow^* \gamma\alpha\beta(A_3, \langle\overline{\alpha}^R\rangle)\overline{\gamma}^R \Longrightarrow^* \gamma\alpha\beta\overline{\alpha}^R\overline{\gamma}^R.$$

This means that $\gamma\alpha\beta\overline{\alpha^R\gamma^R}$ is generated by G'. The converse inclusion, namely $L(G') \subseteq L_r(\alpha) \rightharpoonup_k$, is left to the reader. The first part of the proof is complete as soon as we recall that the class of linear languages is closed under morphisms.

Now, for a linear language L generated by a linear grammar $G = (N, V, S, P)$ we construct a regular language R and a morphism h such that $L = h(R \rightharpoonup_k)$. Without loss of generality we may assume that $V \cap \overline{V} = \emptyset$. Let $G' = (N, V \cup \overline{V} \cup \{\#, \overline{\#}, \$\}, S, P')$ be the right-linear grammar having the set P' defined as follows:

$$P' = \{A \to x\overline{y}^R B \mid A \to xBy \in P\} \cup \{A \to x\#^k\$\overline{\#}^k \mid A \to x \in P\}.$$

The regular language R we are looking for is exactly the language generated by G'. Clearly, every word $w \in R \rightharpoonup_k$ is of the form

$$w = x_1\overline{y_1}^R x_2\overline{y_2}^R \dots x_p\overline{y_p}^R x_{p+1}\#^k\$\overline{\#}^k\overline{x_{p+1}}^R y_p\overline{x_p}^R \dots y_2\overline{x_2}^R y_1\overline{x_1}^R$$

for some $p \geq 0$ and $x_1x_2 \dots x_px_{p+1}y_p \dots y_2y_1 \in L$. We now define the morphism $h : (V \cup \overline{V} \cup \{\#, \overline{\#}, \$\})^* \longrightarrow V^*$ by $h(a) = a$ for any $a \in V$ and $h(b) = h(\#) = h(\$) = \varepsilon$ for any $b \in \overline{V}$, which completes the proof. □

As an immediate consequence we have:

Theorem 3.2. *The hairpin completion of a regular language is not necessarily regular but always linear.*

Proposition 3.3. *1. For any $k \geq 1$, there is a context-free language L such that $L \rightharpoonup_k$ is not context-free.*

2. For any $k \geq 1$ and any context-free language L, $L \rightharpoonup_k$ is context-sensitive.

Proof. 1. Let $L = \{waw^Rb^kc\overline{b^k} | w \in \{0,1\}\}$; clearly, $L \rightharpoonup_k = \{waw^Rb^kc\overline{b^kwaw^R}\}$. We informally define a finite transducer working as follows:

- The first segment consisting of $0, 1$ of the input word, as well as the next a, remain unchanged.
- Then, the transducer removes all symbols until an occurrence of b is met before a symbol in the alphabet $\{0, 1\}$.
- Now, the transducer copy the segment of $0, 1$ until an a is read.
- The rest of the input word is removed.

The reader can easily write the formal details of this transducer. By applying this transducer to $L \rightharpoonup_k$, one gets $\{waw \mid w \in 0, 1\}$ which is a well-known non-context-free language.

2. The second assertion follows immediately by showing that $L \rightharpoonup_k \in NSPACE(n)$. □

We infer as above that:

Theorem 3.4. *The hairpin completion of a context-free language is not necessarily context-free but always context-sensitive.*

It is known that the class of context-sensitive languages is a space complexity class, by the above proof (see also the proof of Theorem 4.1) we can state that such classes are closed under hairpin completion.

Theorem 3.5. *$NSPACE(f(n))$, where $f(n) \geq \log n$ is a space-constructible function, is closed under hairpin completion.*

The case of time complexity classes is slightly different, namely:

Proposition 3.6. *1. If $L \in NTIME(f(n))$, then $L \rightarrow_k \in NTIME(nf(n))$ for any $k \geq 1$.*
2. If $L \in DTIME(f(n))$, then $L \rightarrow_k \in DTIME(nf(n))$ for any $k \geq 1$.

Proof. Let $k \geq 1$, $L \subseteq V^*$, and $w \in V^*$. We prefer to give the argument in the form of a recognizing algorithm for the language $L \multimap_k$ which can be easily implemented on a Turing machine.

Algorithm 3.7.
$i := 1$;
while $(i + k + 1 \leq n - i - k)$
  if $(w[1..i+k] = \overline{w[n-k-i+1..n]^R})$ and $(w[1..n-i] \in L)$
    then $Output:\ w \in L \multimap_k$; halt
    else $i := i + 1$
  endif
endwhile
$Output:\ w \notin L \multimap_k$

In this algorithm, $w[i..j]$ denotes the subword of w starting at position i and ending at position j, $1 \leq i \leq j \leq |w|$. Since a similar recognizer can be designed for the language $L \multimap_k$, we are done. □

As an immediate consequence, we state:

Theorem 3.8. *Both classes* **P** *and* **NP** *are closed under hairpin completions.*

We finish this section by pointing out two open problems:

1. Is it decidable whether or not the hairpin completion of a given regular language is regular?

2. Is the n factor needed in Proposition 3.6?

4 Iterated Hairpin Completion

Theorem 4.1. *$NSPACE(f(n))$, where $f(n) \geq \log n$ is a space-constructible function, is closed under iterated hairpin completion.*

Proof. Let us consider the following recursive boolean function which determines whether or not a given word x is in $L(\rightarrow_k)^*$:

Algorithm 4.2.

Function Membership$(x, L(\rightarrow_k)^*)$;
Membership:=false;
if $x \in L$ then Membership:=true; endif; halt;
if $(|x| \leq 2k)$ and $(x \notin L)$ then halt; endif;
choose nondeterministically a decomposition $x = \gamma\alpha\beta\overline{\alpha}^R\overline{\gamma}^R$, with $\beta\gamma \neq \varepsilon$ and $|\alpha| = k$;
if (Membership$(\gamma\alpha\beta\overline{\alpha}^R, L(\rightarrow_k)^*)$ or Membership$(\alpha\beta\overline{\alpha}^R\overline{\gamma}^R, L(\rightarrow_k)^*)$)
then Membership:=true; halt; endif;

This function can clearly be implemented on an off-line nondeterministic (multi-tape) Turing machine in $f(n)$ space provided that L is accepted by an off-line nondeterministic (multi-tape) Turing machine in $f(n)$ space.

Note that $\log n$ is needed in order to store the left- and right-hand border of the current subword within the input word. By finite state one can keep track of whether or not this subword is complemented. □

By Theorem 3.2, the hairpin completion of a regular language is not necessarily regular but always linear. A natural question regards the iterated hairpin completion of a regular language. Is this language regular or still linear?

Proposition 4.3. *For any $k \geq 1$, the iterated k-hairpin completion of a regular language is not necessarily context-free.*

Proof. Let $L = \{a^k b\overline{a}^k c^n \overline{a}^k \mid n \geq 1\}$ be a regular language, where a, b, c are letters. It is easy to note that

$$(L(\rightarrow_k)^*) \cap R = \{a^k\overline{c}^n a^k\overline{c}^n a^k b\overline{a}^k c^n\overline{a}^k \mid n \geq 1\},$$

provided that R is the regular language $R = \{a^k\overline{c}^n a^k\overline{c}^m a^k b\overline{a}^k c^p\overline{a}^k \mid n, m, p \geq 1\}$. As $(L(\rightarrow_k)^*) \cap R$ is obviously not context-free, we are done. □

However the iterated k-hairpin completion of a regular language is always recognizable in polynomial time.

Theorem 4.4. *If L is a regular language, then $L(\rightarrow_k)^* \in \mathbf{P}$ for any $k \geq 1$.*

Proof. The statement follows from Theorem 4.1 and the inclusion $NSPACE(\log n) \subseteq \mathbf{P}$. □

The following final problem which remains unsolved naturally arises: Can the last theorem be extended to any language in $\mathbf{P}$? In other words, is $\mathbf{P}$ closed under iterated hairpin completion?

References

[1] P. Bottoni, G. Mauri, P. Mussio, Gh. Păun, Grammars working on layered strings, *Acta Cybernetica*, 13(1998), 339–358.

[2] P. Bottoni, A. Labella, V. Manca, V. Mitrana, Superposition based on Watson-Crick-like complementarity, *Theory of Computing Systems* (to appear).

[3] J. Castellanos, V. Mitrana, Some remarks on hairpin and loop languages, *Words, Semigroups, and Tranlations*, (M. Ito, Gh. Păun, S. Yu, eds.), World Scientific, Singapore, 2001 47–59.

[4] R. Deaton, R. Murphy, M. Garzon, D.R. Franceschetti, S.E. Stevens, Good encodings for DNA-based solutions to combinatorial problems, *Proc. of DNA-based computers II*, (L.F. Landweber, E. Baum, eds.), DIMACS Series, vol. 44, 1998, 247–258.

[5] R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa, Bidirectional sticker systems, *Third Annual Pacific Conf. on Biocomputing*, Hawaii, 1998 (R.B. Altman, A.K. Dunker, L. Hunter, T.E. Klein, eds.), World Scientific, Singapore, 1998, 535–546.

[6] M. Garzon, R. Deaton, P. Neathery, R.C. Murphy, D.R. Franceschetti, E. Stevens, On the encoding problem for DNA computing, *The Third DIMACS Workshop on DNA-Based Computing*, Univ. of Pennsylvania, 1997, 230–237.

[7] M. Garzon, R. Deaton, L.F. Nino, S.E. Stevens Jr., M. Wittner, Genome encoding for DNA computing, *Proc. Third Genetic Programming Conference*, Madison, MI, 1998, 684–690.

[8] T. Head, Formal language theory and DNA: an analysis of the generative capacity of recombinant behaviors, *Bulletin of Mathematical Biology*, 49(1987), 737–759.

[9] L. Kari, Gh. Păun, G. Rozenberg, A. Salomaa, S. Yu, DNA computing, sticker systems, and universality, *Acta Informatica*, 35, 5(1998), 401–420.

[10] L. Kari, S. Konstantinidis, P. Sosk, G. Thierrin, On hairpin-free words and languages, *Proc. Developments in Language Theory 2005* (C. De Felice, A. Restivo, eds.), LNCS 3572, Springer-Verlag, Berlin, 2005, 296-307.

[11] S. Kobayashi, V. Mitrana, Gh. Păun, G. Rozenberg, Formal properties of PA-matching, *Theoretical Comput. Sci.*, 262, 1-2(2001), 117–131.

[12] Gh. Păun, G. Rozenberg, Sticker systems, *Theoret. Comput. Sci.*, 204(1998), 183–203.

[13] Gh. Păun, G. Rozenberg, and A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, Berlin, 1998, Tokyo, 1999.

[14] Gh. Păun, G. Rozenberg, T. Yokomori, Hairpin languages, *Intern. J. Found. Comp. Sci.* 12, 6(2001), 837–847.

[15] G. Rozenberg, A. Salomaa, Eds., *Handbook of Formal Languages*, 3 volumes, Springer-Verlag, Berlin, Heidelberg, 1997.

[16] K. Sakamoto, H. Gouzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori, and M. Hagiya, Molecular computation by DNA hairpin formation, *Science* 288(2000), 1223–1226.

Daniela Cheptea
Faculty of Mathematics and Computer Science
University of Bucharest
Str. Academiei 14, 010014, Bucharest, Romania
dcheptea@funinf.cs.unibuc.ro

Carlos Martín-Vide
Research Group in Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tarraco 1, 43005, Tarragona, Spain
carlos.martin@urv.net

Victor Mitrana
Faculty of Mathematics and Computer Science
University of Bucharest
Str. Academiei 14, 010014, Bucharest, Romania
and
Research Group in Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tarraco 1, 43005, Tarragona, Spain
mitrana@fmi.unibuc.ro

Decision Problems for CD Grammar Systems and Grammars with Regulated Rewriting

Liliana Cojocaru

Abstract

In this paper we deal with *cooperating distributed grammar systems* (henceforth CD grammar systems) that work under a powerful protocol, recently introduced in [2]. The new protocol is based on the *level of competence* of a component to rewrite a certain number of nonterminals occurring in the underlying sentential form. We tackle the particular case of $\{\leq 1, = 1\}$-competence mode, for which we propose two coverability structures, the *coverability tree* and the *twin coverability tree* associated to the derivations in a CD grammar system working in $\{\leq 1, = 1\}$-competence mode. The coverability tree structure has been adopted from Petri nets theory, and due to the *finiteness property* that it has, it turned out to be a strong tool in solving decision problems. Analysing the structure of the coverability tree and of the twin coverability tree we solve several decision problems such as *emptiness, finiteness, boundedness* and *reachability* problems for the above mentioned competence-based protocol. In [2] it has been proved that there exists a strong relation between CD grammar systems working in $\{\leq 1, = 1\}$-competence mode and *forbidding random context grammars*. Due to this, the decidability results we have found have positive consequences upon several decision problems left open in [10], for several grammars with regulated rewriting, such as *forbidding random context grammars* and *ordered grammars*.

Introduction

A *cooperating distributed grammar system* is a set of Chomsky grammars that work sequentially on a common sentential form according to a specified *protocol of cooperation*. At a given moment only one component is active. The protocol decides which component is active or idle. It gives running and stop conditions depending on the mode of derivation used by the system. The system models the blackboard architecture in cooperative problem solving, in which the blackboard is the common working tape and the components are knowledge sources (agents, processors, abstract computers, etc.). For the formal definition and basic results related to CD grammar systems the reader is referred to [5], [7] and [8].

Recently, a new cooperation protocol, based on the *level of competence* of a component to rewrite a certain number of different nonterminals on a sentential form has been introduced in [2] and [6] with several forerunner related papers [3], [1]. Several variations on the same theme have been considered in [9], in which the author deals with CD grammar systems with start and stop conditions that work in maximal or full competence mode.

In the case of $\{\leq k, = k, \geq k\}$-competence based protocols the common sentential form is divided into pieces of work distributed rewritten by those components of the CD grammar system that are $\{\leq k, = k, \geq k\}$-competent to proceed the derivation, $k \geq 1$. With respect to the cooperation protocol defined in [2], a component is $\leq k$-competent, $= k$-competent or $\geq k$-competent, $k \geq 1$, on a certain sentential form if it is able to rewrite *at most, exactly*, or *at least* k distinct nonterminals occurring in the sentential form, respectively. Once a component starts to be f-competent on a sentential form, it has to continue the derivation as long as it is f-competent, $f \in \{\leq k, = k, \geq k\}$, $k \geq 1$, on the newest sentential form. The formal definition of competence-based protocols is provided in Section 1.

In [2] it is proved that competence-based protocols lead CD grammar systems to work at least at the level of *forbidding random context grammars* for the case of $\{\leq 1, = 1\}$-competence mode, at the level of *ET0L systems* for ≥ 1-competence mode, and at the level of *random context ETOL systems* for $\geq k$-competence mode, $k \geq 2$. Finally, they are as powerful as *random context grammars* for the case of $\{\leq k, = k\}$-competence mode, $k \geq 2$, i.e., they characterize the class of *recursively enumerable languages*.

In this paper we focus on a structural characterization of CD grammar systems that work in $\{\leq 1, = 1\}$-competence mode, by means of the coverability tree-structure associated to the derivations in such a system. This structure has been firstly introduced in [14], under the name of *reachability tree*, in order to study the reachability problem for vector addition systems. It has been adopted afterwards in Petri nets theory under the name of *coverability tree*, where the structure provides a finite representation of every reachable marking in a Petri net. Informally, the coverability tree is a method that consists of the reduction of an infinite structure (as a derivation in a grammar system) to a finite one.

Besides the *coverability tree*, that gives a structural representation of the variation of the number of nonterminals in the underlying sentential form, we introduce also the *twin coverability tree* associated with a coverability tree for a certain CD grammar system that works in $\{\leq 1, = 1\}$-competence mode. The *twin coverability tree*, offers a structural representation of the number of terminal symbols pumped into the underlying sentential form at each step of derivation. The formal definitions of these coverability trees for CD grammar systems working in $\{\leq 1, = 1\}$-competence mode, are presented in Section 2. We give also several examples that illustrate the manner in which we build these tree-structures. We prove in Section 3.1 that these tree structures are well defined, i.e., they give a good representation of CD grammar systems behaviour and that they are always finite. Therefore they can be used as an analysis tool to check properties of grammar systems. Analysing the structure of these trees, we solve in Section 3.2 several decision problems, such as *emptiness, finiteness, boundedness* or *reachability* for CD grammar systems working in $\{\leq 1, = 1\}$-competence mode. Furthermore, in [2] it has been proved, using a constructive method, that the class of languages generated by CD grammar systems that work in the $\{\leq 1, = 1\}$-competence mode includes the class of languages generated by *forbidding random context grammars*. With an affirmative answer for the emptiness and finiteness problems for CD grammar systems that work in the $\{\leq 1, = 1\}$-competence mode we study in Section 4 the above decision problems for several grammars with regulated rewriting, such as *forbidding random context grammars* and *ordered grammars*.

1 CD grammar systems working in competence mode

The aim of this section is to introduce the basic notions that concern CD grammar systems that work in competence mode. Several basic results and a setting of the notations of concepts from the topic covered in this paper will be given, too. For further information the reader is referred to [8] and [12]. Let X be an alphabet, always composed of a finite set of letters, $|X|$ be the cardinal number of the set X, and X^* be the set of all strings over the alphabet X. The empty word is denoted by λ, and the number of occurrences of the symbol $a \in X$ in the string $w \in X^*$ is denoted by $|w|_a$.

Definition 1.1. A *cooperating distributed grammar system* of degree s, is an $(s+3)$-tuple $\Gamma = (N, T, \alpha, P_1, ..., P_s)$, where N and T are disjoint alphabets, the *nonterminal* and the *terminal* alphabet, respectively. $P_1, ..., P_s$ are finite sets of context-free rules over $N \times (N \cup T)^*$, called the system components, and $\alpha \in (N \cup T)^*$ is the system axiom.

Definition 1.2. Let $\Gamma = (N, T, \alpha, P_1, ..., P_s)$ be a CD grammar system and $\mathrm{dom}(P_i) = \{A \in N | A \to z \in P_i\}$ be the domain of the component P_i, $1 \leq i \leq s$. We say that P_i is *k-competent* on a sentential form x, $x \in (N \cup T)^*$, if and only if $|\mathrm{alph}_N(x) \cap \mathrm{dom}(P_i)| = k$, where $\mathrm{alph}_N(x) = \{A \in N | |x|_A > 0\}$.

In the sequel we denote by $\mathrm{clev}_i(x)$ the level of competence of the component P_i on x. The cooperation protocol, based on the capability of a component P_i, $1 \leq i \leq s$, to be $\leq k$-*competent*, $= k$-*competent*, $\geq k$-*competent* on a sentential form x, $x \in (N \cup T)^*$, is defined in [2] as follows.

Definition 1.3. Let $\Gamma = (N, T, \alpha, P_1, ..., P_s)$ be a CD grammar system, $x, y \in (N \cup T)^*$, $1 \leq i \leq s$.
1. We say that y is derived from x in the $\leq k$-*competence mode* of derivation, denoted by $x \Rightarrow_i^{\leq k\text{-comp.}} y$, iff there exists a derivation $x = x_0 \Rightarrow_i x_1 \Rightarrow_i ... \Rightarrow_i x_{m-1} \Rightarrow_i x_m = y$ such that (a) $\mathrm{clev}_i(x_j) \leq k$ for $0 \leq j < m$ and (i) $\mathrm{clev}_i(x_m) = 0$ or (ii) $y \in T^*$, or
(b) $\mathrm{clev}_i(x_j) \leq k$ for $0 \leq j < m$ and $\mathrm{clev}_i(x_m) > k$.
2. We say that y is derived from x in the $= k$-*competence mode* of derivation, denoted by $x \Rightarrow_i^{=k\text{-comp.}} y$, iff there exists a derivation $x = x_0 \Rightarrow_i x_1 \Rightarrow_i ... \Rightarrow_i x_{m-1} \Rightarrow_i x_m = y$ such that (a) $\mathrm{clev}_i(x_j) = k$ for $0 \leq j < m$ and $\mathrm{clev}_i(x_m) \neq k$ or
(b) $\mathrm{clev}_i(x_0) = k$, $\mathrm{clev}_i(x_j) \leq k$ for $1 \leq j \leq m$ and $y \in T^*$.
3. We say that y is derived from x in the $\geq k$-*competence mode* of derivation, denoted by $x \Rightarrow_i^{\geq k\text{-comp.}} y$, iff there exists a derivation $x = x_0 \Rightarrow_i x_1 \Rightarrow_i ... \Rightarrow_i x_{m-1} \Rightarrow_i x_m = y$ such that (a) $\mathrm{clev}_i(x_j) \geq k$ for $0 \leq j < m$ and $\mathrm{clev}_i(x_m) < k$ or
(b) $\mathrm{clev}_i(x_0) \geq k$ and $y \in T^*$.

Let $M = \{\leq k\text{-comp.}, = k\text{-comp.}, \geq k\text{-comp.} | k \geq 1\}$, and let $\Rightarrow^f$ denote $\Rightarrow_i^f$, for some i, $1 \leq i \leq s$, $f \in M$. We denote by $\Rightarrow^{*f}$ the *reflexive* and *transitive closure* of $\Rightarrow^f$.

Definition 1.4. The *language generated* by Γ in f-mode of derivation, $f \in M$, is

$$L_f(\Gamma) = \{w \in T^* | \alpha \Rightarrow^{*f} w\}.$$

The *family of languages* generated by CD grammar systems in f-mode, $f \in M$, is denoted by $\mathcal{L}(\text{CD}, \text{CF}, f)$.

Definition 1.5. The *Parikh vector* associated with a string $x \in X^*$ with respect to the alphabet $X = \{a_1, a_2, ..., a_p\}$ is $\Psi_X(x) = (|x|_{a_1}, |x|_{a_2}, ..., |x|_{a_p})$.

Definition 1.6. Let $\Gamma = (N, T, \alpha, P_1, ..., P_s)$ be a CD grammar system and let $L_f(\Gamma)$ be the language generated by Γ in f-mode, $f \in M$. The set of all sentential forms obtained during terminal derivations of a word $w \in L_f(\Gamma)$, is denoted by

$$S_f(\Gamma, w) = \{x | \alpha \Rightarrow^{*f} x \Rightarrow^{*f} w, w \in L_f(\Gamma)\}.$$

We denote by $S_f(\Gamma)$ the set of all sentential forms obtained during the derivation process of $L_f(\Gamma)$, i.e., $S_f(\Gamma) = \{x | x \in S_f(\Gamma, w), w \in L_f(\Gamma)\}$.

The *sentential form Parikh set* associated to the language $L_f(\Gamma)$, with respect to the nonterminals alphabet N, starting with the axiom α, is the set of all Parikh vectors associated with all possible sentential forms obtained during the derivation process of $L_f(\Gamma)$.

The *sentential form Parikh set* associated with $L_f(\Gamma)$, with respect to the *nonterminal alphabet* N, is denoted by $\Psi_N(S_f(\Gamma)) = \{\Psi_N(x) | x \in S_f(\Gamma)\}$.

The *sentential form Parikh set* associated with $L_f(\Gamma)$, with respect to the *terminal alphabet* T, is analogously defined as $\Psi_T(S_f(\Gamma)) = \{\Psi_T(x) | x \in S_f(\Gamma)\}$.

2 Coverability trees for competence-based protocols

The *coverability tree* is a state-based structure intensively used to solve decision problems for Petri nets. It is built starting from the initial state of the Petri net by exhaustively building the reachable state space. Through the special symbol ω, representing the marking of an unbounded transition node, the method functions also for unbounded nets. Thus, the set of natural numbers $\mathbf{N}$ is extended to the set $\mathbf{N}_\omega = \mathbf{N} \cup \{\omega\}$ defined by the arithmetical rules $\omega + \omega = \omega + a = a + \omega = \omega$, $\omega - a = \omega$, $\omega \cdot a = a \cdot \omega = \omega$ for $a \in \mathbf{N}$ and $a < \omega$.

Furthermore, we consider $\mathbf{N}_{\omega_t} = \mathbf{N} \cup \{\omega_t\}$, where ω_t is a marked ω symbol used in the construction of the coverability tree for CD grammar systems working in $\{\leq 1, = 1\}$-comp.-mode. The addition and multiplication operations are defined for this special symbol as follows: $\omega_t + a = a + \omega_t = a$, $\omega_t + \omega_t = \omega_t$, $\omega_t + \omega = \omega + \omega_t = \omega$, $\omega_t \cdot a = a \cdot \omega_t = \omega_t \cdot \omega_t = \omega_t$, for $a \in \mathbf{N}$, and $\omega_t < 1 \leq a < \omega$. We denote by $\mathbf{N}_{\{\omega,\omega_t\}} = \mathbf{N} \cup \{\omega, \omega_t\}$. Observe that, by definition ω and ω_t cannot be decreased. The above operations defined on $\mathbf{N}_\omega$, $\mathbf{N}_{\omega_t}$ and $\mathbf{N}_{\{\omega,\omega_t\}}$ can be extended in the same way for the space $\mathbf{N}_\omega^m$, $\mathbf{N}_{\omega_t}^m$ and $\mathbf{N}_{\omega,\omega_t}^m$, respectively.

For each $U \in \mathbf{N}_{\omega,\omega_t}^m$ we denote by $U(j)$ the j^{th} place of U. We say that between two vectors $U, V \in \mathbf{N}_{\omega,\omega_t}^m$, holds the relation $U \leq V$, if and only if $U(j) \leq V(j)$, for $1 \leq j \leq m$.

Due to many similarities that exist between distributed grammar systems and Petri nets, the coverability tree-based method has been proved to be a strong decidability tool in the theory of grammar systems, too. It has been already applied for parallel communicating grammar systems in [20], [21], [16] and for cooperating distributed grammar systems in [17], [18]. Our solution in solving decidability problems for CD grammar systems working in $\{\leq 1, = 1\}$-comp.-mode is based on the same approaching.

To have a better picture upon the way in which the coverability tree associated to the derivations in a CD grammar system that work in $\{\leq 1, = 1\}$-comp.-mode is built, we recall that for the $= 1$-comp.-mode a component is not allowed to perform a derivation step if two or more than two different nonterminals occur in the current sentential form. These nonterminals have to be rewritten by other components that are $= 1$-competent on that sentential form, otherwise the derivation cannot continue anymore. However, a component is allowed to keep on rewriting nonterminals as long as it is competent on the newest sentential form. So that once a component becomes $= 1$-competent, it will be active as long as the component is $= 1$-competent on the newest sentential form.

The ≤ 1-comp.-mode works in the same manner, with the difference that this protocol can delay the process of derivation due to the freedom property to not rewrite nonterminals on a given sentential form. This property has consequences only on the length of derivation. It does not change the language, so that we have $\mathcal{L}$(CD,CF,$\leq$ 1-comp)= $\mathcal{L}$(CD,CF, $=$ 1-comp).

Let $\Gamma = (N, T, \alpha, P_1, ..., P_s)$ be a CD grammar system with s components, $s \geq 1$, $N = \{A_1, A_2, ..., A_m\}$ be the ordered set of nonterminal symbols, $|N| = m$, $m \geq 1$, and $T = \{a_1, a_2, ..., a_p\}$ be the ordered set of terminal symbols, $|T| = p$, $p \geq 1$.

Let Ł $= \{1, 2, ..., s\}$ be the set of labels attached to the system components. For each component P_i, $i \in$ Ł, we denote by Ł$_i = \{1, 2, ..., |P_i|\}$ the set of labels attached to productions in P_i, where $|P_i|$ is the number of rules of P_i. We denote by $\mathcal{L}$ the set of all pairs $(i, j) \in \mathbf{N^2}$ such that $i \in$ Ł and $j \in$ Ł$_i$.

To any context-free rule r, of the form $lhs(r) \to rhs(r)$ we attach two vectors $\Delta_r \in (\mathbf{N} \cup \{-1\})^m$, and $\Delta_{r,t} \in \mathbf{N}^p$, defined as:

$$\Delta_r = (|rhs(r)|_{A_1} - |lhs(r)|_{A_1}, |rhs(r)|_{A_2} - |lhs(r)|_{A_2}, ..., |rhs(r)|_{A_m} - |lhs(r)|_{A_m}),$$
$$\Delta_{r,t} = (|rhs(r)|_{a_1}, |rhs(r)|_{a_2}, ..., |rhs(r)|_{a_p}).$$

The sequential behaviour of Γ, working in $\{\leq 1, = 1\}$-comp.-mode, is characterized by the next *firing rules* adopted from Petri net theory (see [19]).

The $\{\leq 1, = 1\}$-*enabling rule:*
The element $\vartheta \in \mathcal{L}$, $\vartheta = (i, j)$, $i \in$ Ł, $j \in$ Ł$_i$, is *enabled at the vector* $V \in \mathbf{N}^m_{\omega,\omega_t}$, abbreviated $V[\vartheta\rangle_\Gamma$, if $V(q) \geq |lhs(r)|_{A_q}$, $A_q \in \text{dom}(P_i)$, $1 \leq q \leq m$, where r is the production from P_i labeled by j, and there is no other place q' in V, $q \neq q'$, such that $V(q') \geq |lhs(r')|_{A'_q}$, $A'_q \in \text{dom}(P_i)$, $r' \in P_i$.

The $\{\leq 1, = 1\}$-*computing rule:*
If $V[\vartheta\rangle_\Gamma$ then ϑ may *occur* yielding a new vector V', computed with respect to the rule $r \in P_{\vartheta(1)}$, labeled by $\vartheta(2)$, nondeterministically chosen from $P_{\vartheta(1)}$, with respect to the $\{\leq 1, = 1\}$-*enabling rule*, to rewrite the nonterminal A_q. V' is defined by $V' = V + \Delta_r$. We abbreviate this computing rule by $V[\vartheta\rangle_\Gamma V'$.

Note that, due to the inequalities that characterize the ω_t symbol, i.e., $\omega_t < 1 \leq a < \omega$, the relation $V(q) \geq |lhs(r)|_{A_q}$, $A_q \in \text{dom}(P_i)$, r a production in P_i, $1 \leq i \leq s$, never holds for $V(q) = \omega_t$. That is why enabling rules depend only on those values of $V(q)$ that are in $\mathbf{N}_\omega$. Furthermore, if more than one component P_i, $1 \leq i \leq m$, satisfies the $\{\leq 1, = 1\}$-computing rule, then at a vector $V \in \mathbf{N}^m_{\omega,\omega_t}$ more than one vector $\vartheta \in \mathcal{L}$ might be enabled.

Definition 2.1. The *coverability tree* attached to a grammar system Γ working in $\{\leq 1, = 1\}$-comp.-mode, is a rooted tree $\mathcal{B}(\Gamma) = (\mathcal{V}, \mathcal{E})$, in which the vertices are elements in $\mathbf{N}^m_{\omega,\omega_t}$ and the edges are elements in $\mathcal{L}$. The set of nodes and edges are defined as follows:

1) The root v_0 is labeled by the Parikh vector, $\Psi_N(\alpha)$ associated to the system axiom α, such that there is at least one element $\vartheta \in \mathcal{L}$, $\{\leq 1, = 1\}$-enabled at the vector $\Psi_N(\alpha)$. The children of v_0 are nodes labeled by those vectors $\Psi_N(\alpha')$ such that $\Psi_N(\alpha)[\vartheta\rangle_\Gamma \Psi_N(\alpha')$ and $\Psi_N(\alpha') = \Psi_N(\alpha) + \Delta_r$, where r is the rule labeled by $\vartheta(2)$ from $P_{\vartheta(1)}$. In this case ϑ is the label of the edge from v_0 to its child labeled by $\Psi_N(\alpha')$.

2) For any $v \in \mathcal{V}$, $v \neq v_0$, labeled by a vector $\Psi_N(\bar{\alpha}) \in \mathbf{N}^m_{\omega,\omega_t}$, there is at least one element $\vartheta \in \mathcal{L}$, $\{\leq 1, = 1\}$-enabled at the vector $\Psi_N(\bar{\alpha})$. Let us denote by $L \subseteq \mathcal{L}$ the set of all these elements. Let v_p be the parent node of v, and $\vartheta_p \in \mathcal{L}$ be the label of the edge from v_p to v. We consider $L_1 = \{\vartheta | \vartheta \in L, \vartheta(1) = \vartheta_p(1)\}$ and $L_2 = \{\vartheta | \vartheta \in L, \vartheta(1) \neq \vartheta_p(1)\}$. Then $L = L_1 \cup L_2$ and $L_1 \cap L_2 = \emptyset$. The children of v are nodes labeled by those vectors $\Psi_N(\bar{\alpha}')$ such that $\Psi_N(\bar{\alpha})[\vartheta\rangle_\Gamma \Psi_N(\bar{\alpha}')$ and $\vartheta \in L_1$ if $L_1 \neq \emptyset$, or $\vartheta \in L_2$ if $L_1 = \emptyset$, and there is no other node $\bar{v}$ on the path from v_0 to v labeled by $\Psi_N(\bar{\alpha}')$ and yielded by ϑ. We say that ϑ is the label of the edge from v to its child labeled by $\Psi_N(\bar{\alpha}')$. Furthermore,

$$\Psi_N(\bar{\alpha}')(j) = \begin{cases} \omega, & \begin{cases} \text{-if there exists a node } \bar{v} \text{ on the path from } v_0 \text{ to } v \text{ labeled by} \\ \Psi_N(\hat{\alpha}) \text{ such that } \Psi_N(\hat{\alpha}) \leq \Psi_N(\bar{\alpha}') \text{ and } \Psi_N(\hat{\alpha})(j) < \Psi_N(\bar{\alpha}')(j); \\ \text{-if there exists } j',\ 1 \leq j' \leq m,\ j' \neq j, \text{ such that } \Psi_N(\bar{\alpha})(j') = \omega, \\ \Delta_r(j) > 0, \text{ and } r \in P_{\vartheta(1)}, \text{ labeled by } \vartheta(2), \text{ rewrites } A_{j'}; \end{cases} \\ \omega_t, & \text{if } \Psi_N(\bar{\alpha})(j) = \omega, \text{ and } \Delta_r(j) = -1; \\ \Psi_N(\bar{\alpha})(j) + \Delta_r(j), & \text{otherwise.} \end{cases}$$

Note that the above coverability tree attached to a CD grammar system Γ, simulates the work of Γ, in the $\{\leq 1, = 1\}$-comp.-mode, as follows. The root is labeled by the Parikh vector attached to the axiom α. Each child of the root is a node labeled by the Parikh vector attached to the sentential form obtained from α after the application of only one rule from the component that is $\{\leq 1, = 1\}$-competent on α. Once a component starts to be $\{\leq 1, = 1\}$-competent on a sentential form, it continues the derivation as long as it is $\{\leq 1, = 1\}$-competent on the newest sentential form. This characterization is given by the $\vartheta(1) = \vartheta_p(1)$ condition that is checked at each new node v added in the coverability tree, where v_p is the parent node of v.

In the case that at a given point of derivation an arbitrary number of nonterminals are "pumped" into the new sentential form, by nonlinear context-free rules, we use the ω-notation in order to mark in the Parikh vector associated to the underlying sentential form, those places with the above property. We call the new resulting vector an ω-Parikh vector. Through this "ω-marking", any node $v \in \mathcal{V}$, different from the root, is labeled by the Parikh vector or by the ω-Parikh vector associated with the corresponding sentential form obtained during the generative process.

A place $V(q)$ in an ω-Parikh vector $V \in \mathbf{N}^m_\omega$, yielded by an element $\vartheta \in \mathcal{L}$, that has the ω-value cannot be decreased. However, due to the $\{\leq 1, = 1\}$-competence protocol

the nonterminals A_q settled at the place $V(q)$, can be rewritten by rules of the component $P_{\vartheta(1)}$, of the form $A_q \to c$, $c \in T^*$, $A_q \to \lambda$, or $A_q \to x$, $x \in (N \cup T)^*$, such that $\text{alph}_N(x) \cap \text{dom}(P_{\vartheta(1)}) = \emptyset$, until all nonterminals A_q are rewritten by such rules. In order to overcome this phenomenon we associate an emptiness limit ω_t to each place $V(q)$ in an ω-Parikh vector $V \in \mathbf{N}^m_\omega$, that is entirely rewritten. The idea to introduce this limit comes from the fact that once the component $P_{\vartheta(1)}$ is $= 1$-competent on a sentential form it continues the derivation until it loses its competence. As the sentential form x does not contain any symbol from $\text{dom}(P_{\vartheta(1)})$ the above rules can be applied until A_q is fully rewritten. The ω_t symbol is seen as an emptiness limit for those places that do not hold any nonterminal symbol anymore.

Leaves, in the above coverability tree, are those nodes that end or block the derivation process, i.e., they are labeled by null vectors or by vectors from $\mathbf{N}^m_{\omega_t}$ (because the underlying sentential form contains only terminal symbols), or by vectors that have at least two places that correspond to nonterminals that can be rewritten by only one component, so that the derivation in the $\{\leq 1, = 1\}$-comp.-mode cannot continue anymore. In other words no element of $\mathcal{L}$ can be enabled in these nodes. Finally, the last case stands for those nodes v, labeled by a vector V for which the pair (ϑ, V'), $\vartheta \in \mathcal{L}$, V,$V' \in \mathbf{N}^m_{\omega,\omega_t}$, where $V[\vartheta\rangle V'$, occurs already on the path from the root to v, so that the node v cannot have any children due to the item 2) in the definition of the coverability tree. We call these last types of leaves *cutting points*, because they are in fact cutting points in loops that appear during the derivation process.

To each coverability tree $\mathcal{B}(\Gamma) = (\mathcal{V}, \mathcal{E})$, associated to the derivations in a CD grammar system, Γ that works in $\{\leq 1, = 1\}$-comp.-mode, we associate a *twin coverability tree*, denoted as $\mathcal{B}_t(\Gamma) = (\mathcal{V}_t, \mathcal{E}_t)$, built with respect to the terminal symbols pumped into the underlying sentential form at each step of derivation performed by Γ. Informally, the twin coverability tree is built in the same manner as the coverability tree. The difference is that each node introduced in the twin coverability tree is labeled by a vector in $\mathbf{N}^p_\omega$ that characterizes the number of terminals pumped at each step of derivation into the underlying sentential form. Because in the twin coverability tree we take into account also the number of terminals pumped into the sentential form when the system enters in a loop[1], the twin coverability tree might have more nodes and edges than the coverability tree. Formally, the twin coverability tree is defined as follows.

Definition 2.2. The *twin coverability tree* attached to a *coverability tree* for a CD grammar system Γ, working under the $\{\leq 1, = 1\}$-comp.-mode protocol, is a rooted tree $\mathcal{B}_t(\Gamma) = (\mathcal{V}_t, \mathcal{E}_t)$, in which the vertices are elements in $\mathbf{N}^p_\omega$ and the edges are elements in $\mathcal{L}$. The set of nodes and edges are defined as follows:

1)The root v_0 is labeled by the Parikh vector $\Psi_T(\alpha)$ attached to the system axiom α, such that there is at least one element $\vartheta \in \mathcal{L}$, $\{\leq 1, = 1\}$-enabled at the vector $\Psi_N(\alpha)$. The children of v_0 are nodes labeled by those vectors $\Psi_T(\alpha')$ defined as $\Psi_T(\alpha') = \Psi_T(\alpha) + \Delta_{r,t}$ where r is the rule labeled by $\vartheta(2)$ from $P_{\vartheta(1)}$, and ϑ is the label of the edge from v_0 to the child node labeled by $\Psi_T(\alpha')$ such that $\Psi_N(\alpha)[\vartheta\rangle_\Gamma \Psi_N(\alpha')$ and $\Psi_N(\alpha') = \Psi_N(\alpha) + \Delta_r$.

[1]This phenomenon is marked in the coverability tree by cutting points.

2) For any $v \in \mathcal{V}_t$, $v \neq v_0$, labeled by a vector $\Psi_T(\bar{\alpha}) \in \mathbf{N}_\omega^p$, there exists at least one element $\vartheta \in \mathcal{L}$, $\{\leq 1, = 1\}$-enabled at the vector $\Psi_N(\bar{\alpha})$. Let us denote by $L \subseteq \mathcal{L}$ the set of all these elements. Next, we consider the same sets L_1 and L_2, as we have settled for the coverability tree, more exactly $L_1 = \{\vartheta | \vartheta \in L, \vartheta(1) = \vartheta_p(1)\}$ and $L_2 = \{\vartheta | \vartheta \in L, \vartheta(1) \neq \vartheta_p(1)\}$. Then $L = L_1 \cup L_2$ and $L_1 \cap L_2 = \emptyset$. The children of v are nodes labeled by those vectors $\Psi_T(\bar{\alpha}')$ defined by:

$$\Psi_T(\bar{\alpha}')(j) = \begin{cases} \omega, & \text{if there exists } j',\ 1 \leq j' \leq m, \text{ such that } \Psi_N(\bar{\alpha})(j') = \omega, \\ & \Delta_{r,t}(j) > 0 \text{ and } r \in P_{\vartheta(1)}, \text{ labeled by } \vartheta(2), \text{ rewrites } A_{j'}; \\ \Psi_T(\bar{\alpha})(j) + \Delta_{r,t}(j), & \text{otherwise;} \end{cases}$$

where ϑ is enabled at $\Psi_N(\bar{\alpha})$, $\Psi_N(\bar{\alpha})[\vartheta\rangle_\Gamma \Psi_N(\bar{\alpha}')$, and r is the rule from $P_{\vartheta(1)}$, labeled by $\vartheta(2)$. Furthermore, $\vartheta \in L_1$ if $L_1 \neq \emptyset$, and $\vartheta \in L_2$ if $L_1 = \emptyset$, and there is no other node $\bar{v}$ on the path from v_0 to v in the coverability tree, labeled by $\Psi_N(\bar{\alpha}')$ and yielded by ϑ.

In the case that there exists one node $\bar{v}$ on the path from v_0 to v, labeled by $\Psi_N(\bar{\alpha}')$ and yielded by ϑ in the initial coverability tree, then a new node $\bar{v}'$ is added in the twin coverability tree attached to the coverability tree (which is the case of a cutting point in the coverability tree). It is labeled by $\Psi_T(\bar{\alpha}') \in \mathbf{N}_\omega^p$ and it is defined by:

$$\Psi_T(\bar{\alpha}')(j) = \begin{cases} \omega, & \text{if there exists at least one vector } \vartheta \in \mathcal{L} \text{ enabled at one of} \\ & \text{the vectors } V \in \mathbf{N}_{\omega,\omega_t}^m \text{ that label the nodes from the path} \\ & v,\ \bar{v} \text{ to } v, \textit{ i.e.}, v\bar{v}...v, \text{ in the coverability tree, such that} \\ & \Delta_{r,t}(j) > 0, \text{ where } r \text{ is the rule from } P_{\vartheta(1)} \text{ labeled by } \vartheta(2); \\ \Psi_T(\bar{\alpha})(j), & \text{otherwise (that is the case when no terminal is pumped} \\ & \text{into the sentential form during the execution of the loop).} \end{cases}$$

The label of the edge $v\bar{v}'$, is the element ϑ enabled at $\Psi_N(\bar{\alpha})$.

Next we give two examples that illustrate the way in which the coverability and the twin coverability trees are built.

Example 2.3. Let $\Gamma = (\{A, B, C, D, A', B', D'\}, \{a, b, c\}, \alpha, P_1, P_2, P_3)$ be a CD grammar system that works in $= 1$-comp.-mode, that has:

$$P_1 = \{\underline{A \to aA'b}_1, \underline{A \to ab}_2, \underline{B \to CB'}_3, \underline{B \to C}_4\},$$
$$P_2 = \{\underline{D \to D'}_1, \underline{D' \to AB}_2, \underline{A' \to A}_3, \underline{B' \to B}_4\},$$
$$P_3 = \{\underline{B \to c}_1, \underline{C \to B}_2\}.$$

Note that each time two (or more than two) nonterminals of the set $\{D, A', B', D'\}$ appear together on the same axiom or sentential form, the derivation in the $= 1$-comp.-mode cannot be continued by the component P_2. The derivation will be taken by one of the components P_1 or P_3, in the case that only one nonterminal from $\text{dom}(P_1)$ or $\text{dom}(P_3)$ occurs in the sentential form, otherwise the derivation will be blocked on that branch of derivation. The same remark holds for the component P_1 and P_3 each time A, B or B, C appear together in an axiom or sentential form, respectively. Next, let us consider $\alpha = AD$.

The coverability tree attached to Γ is $\mathcal{B}(\Gamma) = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E}= \{v_0v_b, v_0v_1, v_1v_2, v_2v_3, v_3v_4, v_4v_{01}, v_4v_5, v_5v_6, v_6v_{02}, v_0v_7, v_7v_8, v_8v_9, v_9v_{10}, v_{10}v_{11}, v_{11}v_{12}, v_{12}v_{13}, v_{13}v_{14}, v_{14}v_{03}, v_{14}v_{15}, v_{15}v_{16}, v_{16}v_{04}, v_{10}v_{17}, v_{17}v_{18}, v_{18}v_{05}, v_{18}v_{19}, v_9v_{20}, v_{20}v_{06}, v_{20}v_{21}, v_{21}v_{22}, v_{22}v_{07}\}$ and $\mathcal{V}= \{v_0, v_b, v_1, ..., v_{22}, v_{01}, ..., v_{07}\}$. The nodes are labeled by:

$v_0 = (1,0,0,1,0,0,0)$, $v_{11} = (0,0,0,0,2,0,0)$, $v_b = (0,0,0,1,1,0,0)$,
$v_1 = (0,0,0,1,0,0,0)$, $v_{10} = v_{12} = (1,0,0,0,1,0,0)$, $v_7 = (1,0,0,0,0,0,1)$,
$v_2 = (0,0,0,0,0,0,1)$, $v_9 = v_{13} = (2,0,0,0,0,0,0)$, $v_8 = (2,1,0,0,0,0,0)$,
$v_3 = (1,1,0,0,0,0,0)$, $v_4 = v_6 = v_{14} = v_{16} = v_{18} = v_{20} = v_{22} = (1,0,0,0,0,0,0)$,
$v_5 = v_{15} = v_{17} = v_{19} = v_{21} = (0,0,0,0,1,0,0)$, $v_{01} = ... = v_{07} = (0,0,0,0,0,0,0)$.

The edges are labeled by: $v_0v_b = v_4v_5 = v_9v_{10} = v_{10}v_{11} = v_{14}v_{15} = v_{18}v_{19} = v_{20}v_{21} = (1,1)$, $v_0v_1 = v_4v_{01} = v_6v_{02} = v_9v_{20} = v_{10}v_{17} = v_{13}v_{14} = v_{14}v_{03} = v_{16}v_{04} = v_{18}v_{05} = v_{20}v_{06} = v_{22}v_{07} = (1,2)$, $v_1v_2 = v_0v_7 = (2,1)$, $v_2v_3 = v_7v_8 = (2,2)$, $v_5v_6 = v_{11}v_{12} = v_{12}v_{13} = v_{15}v_{16} = v_{17}v_{18} = v_{21}v_{22} = (2,3), v_3v_4 = v_8v_9 = (3,1)$.

Observe that, in the node $v_b = (0,0,0,1,1,0,0)$ none of elements in $\mathcal{L}$ can be enabled, because $v(4) \geq |lhs(r)|_{A_4}$, $v(5) \geq |lhs(r')|_{A_5}$, and $A_4 = D$, $A_5 = A' \in \mathrm{dom}(P_2)$, $r, r' \in P_2$. The node v_6, is a cutting node, because the node v_5, is yielded for the second time in the path from the root v_0 to v_6, at $\vartheta = (1,1)$ enabled at vector $(1,0,0,0,0,0,0)$ attached to v_6. Due to the same phenomenon the nodes v_{13}, v_{16}, v_{19}, and v_{22} are cutting points, too. The language is generated as follows:

$AD \Rightarrow_{(1,1)} \underline{aA'bD}_{v_b}$;
$AD \Rightarrow_{(1,2)} abD \Rightarrow_{(2,1)} abD' \Rightarrow_{(2,2)} abAB \Rightarrow_{(3,1)} abAc \Rightarrow_{(1,1)} abaA'bc \Rightarrow_{(2,3)} abaAbc \Rightarrow_{(1,1)} ... \Rightarrow_{(1,2)} aba^mb^mc$;
$AD \Rightarrow_{(2,1)} AD' \Rightarrow_{(2,2)} AAB \Rightarrow_{(3,1)} AAc \Rightarrow_{(1,1)} aA'bAc \Rightarrow_{(1,1)} aA'baA'bc \Rightarrow_{(2,3)} ... \Rightarrow_{(1,2)} a^nb^na^mb^mc$.

The twin coverability tree attached to Γ is $\mathcal{B}_t(\Gamma) = (\mathcal{V}_t, \mathcal{E}_t)$, $\mathcal{V}_t = \mathcal{V} \cup \{v'_5, v'_{10}, v'_{15}, v'_{18}, v'_{21}\}$ and $\mathcal{E}_t = \mathcal{E} \cup \{v_6v'_5, v_{16}v'_{15}, v_{13}v'_{10}, v_{19}v'_{18}, v_{22}v'_{21}\}$. All the new looping edges are labeled by the same vector $(1,1)$, with the exception of $v_{19}v'_{18} = (2,3)$. The nodes are labeled by:

$v_0 = v_7 = v_8 = (0,0,0)$, $v_4 = v_{10} = v_{20} = (1,1,1)$, $v_9 = (0,0,1)$,
$v_b = v_1 = v_2 = v_3 = (1,1,0)$, $v'_5 = v'_{10} = v'_{15} = v'_{18} = v'_{21} = (\omega, \omega, 1)$,
$v_5 = v_6 = v_{11} = v_{12} = v_{13} = v_{17} = v_{18} = v_{01} = v_{06} = v_{21} = v_{22} = (2,2,1)$,
$v_{14} = v_{19} = v_{02} = v_{05} = v_{07} = (3,3,1)$, $v_{15} = v_{16} = v_{03} = (4,4,1)$, $v_{04} = (5,5,1)$.

Note that, because $v_6v'_5$, $v_{16}v'_{15}$, $v_{13}v'_{10}$, $v_{22}v'_{21}$, are looping edges, labeled by $(1,1)$ that use the first rule of the first component, they pump an unbounded number of terminals into the newest sentential form. That is why v'_5, v'_{10}, v'_{15}, v'_{21} are marked by an ω-vector. The node v'_{18} is also labeled by an ω-vector, because the pumping process is performed during the execution of the loop. The language is $L_{=1\text{-comp.}}(\Gamma) = \{a^nb^na^mb^mc | n \geq 1, m \geq 1\}$.

Example 2.4. In order to have an example for a coverability tree with ω and ω_t-notations, we consider the following grammar, working in = 1-comp.-mode:

$\Gamma = (\{A, B, C, D, A', B', D'\}, \{a, b, c\}, \alpha, P_1, P_2, P_3)$
$P_1 = \{\underline{A \to aA'b}_1, \underline{A \to ab}_2, \underline{B \to CB'}_3, \underline{B \to C}_4\}$,
$P_2 = \{\underline{D \to D'}_1, \underline{D' \to AB}_2, \underline{A' \to A}_3, \underline{B' \to B}_4\}$,
$P_3 = \{\underline{B \to c}_1, \underline{C \to c}_2\}$.

For the axiom $\alpha = B$ the coverability tree, following the ω and ω_t-notations, is given by $\mathcal{B}(\Gamma) = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}= \{v_0, v_1, ..., v_{15}, v_{01}, ..., v_{07}\}$ and $\mathcal{E}= \{v_0v_{01}, v_0v_1, v_1v_{02}, v_0v_2, v_2v_3, v_3v_4, v_4v_5, v_5v_{03}, v_3v_6, v_6v_7, v_7v_8, v_8v_9, v_6v_{10}, v_{10}v_{11}, v_{11}v_{12}, v_{12}v_{05}, v_2v_{13}, v_{13}v_{14}, v_{11}v_{04}, v_{14}v_{06}, v_{14}v_{15}, v_{15}v_{07}\}$. The vectors[2] that label the nodes are:

$v_0 = (0,1,0,0,0,0,0)$, $v_2 = (0,0,1,0,0,1,0)$, $v_3 = (0,1,1,0,0,0,0)$,
$v_4 = (0,0,2,0,0,0,0)$, $v_6 = (0,0,\omega,0,0,1,0)$, $v_9 = (0,0,\omega_t,0,0,0,0)$,
$v_7 = (0,1,\omega,0,0,0,0,)$, $v_8 = (0,0,\omega,0,0,0,0)$ $v_{11} = (0,1,\omega_t,0,0,0,0)$,
$v_{04} = (0,0,\omega_t,,0,0,0,0)$ $v_{10} = (0,0,\omega_t,0,0,1,0)$, $v_{13} = (0,0,0,0,0,1,0)$,
$v_1 = v_5 = v_{12} = v_{15} = (0,0,1,0,0,0,0)$, $v_{14} = (0,1,0,0,0,0,0)$,
$v_{01} = ... = v_{07} = (0,0,0,0,0,0,0)$.

Edges are labeled by:

$v_0v_2 = v_3v_6 = (1,3)$, $v_0v_1 = v_3v_4 = v_7v_8 = v_{11}v_{12} = v_{14}v_{15} = (1,4)$,
$v_2v_3 = v_6v_7 = v_{10}v_{11} = v_{13}v_{14} = (2,4)$, $v_0v_{01} = v_{11}v_{04} = v_{14}v_{06} = (3,1)$,
$v_1v_{02} = v_4v_5 = v_5v_{03} = v_8v_9 = v_6v_{10} = v_2v_{13} = v_{15}v_{07} = v_{12}v_{05} = (3,2)$.

The twin coverability tree attached to Γ is $\mathcal{B}_t(\Gamma) = (\mathcal{V}_t,\mathcal{E}_t)$, where $\mathcal{V}_t = \mathcal{V} \cup \{v_2', v_6', v_2''\}$ and $\mathcal{E}_t = \mathcal{E} \cup \{v_{11}v_2', v_7v_6', v_{14}v_2''\}$. The new edges are labeled by the same vector $(1,3)$. The nodes are labeled by:

$v_0 = v_1 = v_2 = v_3 = v_4 = v_6 = v_7 = v_8 = v_6' = (0,0,0)$,
$v_{01} = v_{02} = v_5 = v_{13} = v_{14} = v_{15} = (0,0,1)$,
$v_9 = v_{10} = v_{11} = v_{12} = v_{04} = v_{05} = v_2' = v_2'' = (0,0,\omega)$,
$v_{03} = v_{06} = v_{07} = (0,0,2)$.

The language in this case is: $L_{=1\text{-comp.}}(\Gamma) = \{c^n | n \geq 1\}$.

3 Analysis upon the coverability tree

In this section we prove that the coverability tree and the twin coverability tree associated with the derivations in a CDGS working in $\{\leq 1, = 1\}$-comp.-mode, are *finite* and *well defined*. The *finiteness* property of these trees has significant consequences on several decision problems, such as *emptiness, finiteness, boundedness* or *reachability* problems for CD grammar systems working in $\{\leq 1, = 1\}$-comp.-mode.

3.1 Coverability tree properties

Theorem 3.1. *For any CD grammar system Γ, working in $\{\leq 1, = 1\}$-comp.-mode, the coverability tree attached to Γ is a finite tree, and it can be effectively constructed.*

Proof. To prove this statement we use the same method, based on the König's Lemma, as in [19]. Each node in the coverability tree attached to a CD grammar system that works in $\{\leq 1, = 1\}$-comp.-mode, can have only a finite number of children. This holds due to the fact that $|\mathcal{L}|$ is finite, and for each node in the tree an element from $\mathcal{L}$ can be enabled at a vector $V \in \mathbf{N}^m_{\omega,\omega_t}$ only one time. Furthermore, on each path from the root to a leaf in

[2]Some of vectors are equal, but either they are reached on different subtrees, or they are enabled by different elements from $\mathcal{L}$.

the coverability tree, a node labeled by a vector $V \in \mathbf{N}^m_{\omega,\omega_t}$ together with the element $\vartheta \in \mathcal{L}$ enabled at V, is unique.

Let us consider τ an arbitrary path from the root to a given leaf in this tree. The number of vectors that label nodes in this path can be parted into two groups, vectors that can be compared with each other using the $\leq$ relation between two vectors defined at the beginning of Section 2, and incomparable vectors. The first set of vectors is a finite set because the growing of the value on a given position implies an ω-notation that stops this phenomenon. The set of incomparable vectors is finite, too. Otherwise, due to the finite dimension of these vectors, i.e., m, to accomplish the infinity condition at least one value should infinitely increase, but then an ω-notation should be used reducing the cardinality number of this set to a finite one.

Due to the König's Lemma: *Any rooted tree in which each node has a finite number of successors and there is no infinite path emanating from the root is a finite tree.* the above coverability tree is finite. Due to this property the construction of this tree follows directly from Definition 2.1. □

As a consequence of Theorem 3.1 and Definition 2.2 we have.

Corollary 3.2. *For any CD grammar system Γ, working in $\{\leq 1, = 1\}$-comp.-mode, the twin coverability tree associated with the coverability tree attached to Γ is a finite tree, and it can be effectively constructed.*

Theorem 3.3. *Let Γ be a CD grammar system working in $\{\leq 1, = 1\}$-comp.-mode, $\mathcal{B}(\Gamma)$ be the coverability tree attached to Γ, and $L_f(\Gamma)$, $f \in \{\leq 1\text{-comp.}, = 1\text{-comp.}\}$, be the language generated by Γ in f-mode. The sentential form Parikh set associated to $L_f(\Gamma)$ with respect to the nonterminal alphabet N, i.e., $\Psi_N(S_f(\Gamma))$ defined in Definition 1.6, is completely covered by the set of vectors in N^m_ω, that label the nodes in $\mathcal{B}(\Gamma)$, considering $\omega_t = 0$. The same property holds for the sentential form Parikh set associated to $L_f(\Gamma)$ with respect to the terminal alphabet T, i.e., $\Psi_T(S_f(\Gamma))$.*

Proof. This statement is a direct consequence of the way in which the coverability tree simulates the work of a CD grammar system in the $\{\leq 1, = 1\}$-comp.-mode, (see also the explanation given to this simulation that follows Definition 2.1). Due to the arithmetical rules $\omega + \omega = \omega + a = a + \omega = \omega$, $\omega - a = \omega$, $\omega \cdot a = a \cdot \omega = \omega$, $a < \omega$, $a \in \mathbf{N}$, that are valid in $\mathbf{N}_\omega$, the set of all vectors and ω-vectors, that label nodes in the coverability tree, considering $\omega_t = 0$, covers the set of all Parikh vectors associated with sentential forms from $S_f(\Gamma)$. □

Due to Theorem 3.3 all paths in the coverability tree associated to a CD grammar system that works in the $\{\leq 1, = 1\}$-comp.-mode, cover all possible derivations (terminal or not) spent by the system during the generative process of a language. Furthermore, all possible terminal derivations that generate the language $L_f(\Gamma)$, $f \in \{\leq 1\text{-comp.}, = 1\text{-comp.}\}$-mode, can be recovered between paths in the above coverability tree. We conclude that the above tree structures are well defined and they give a good representation of the behaviour of CD grammar systems working in $\{\leq 1, = 1\}$-comp.-mode. Therefore, these structures can be successfully used as an analising tool in solving decision problems for these systems.

3.2 Decidability results

Due to Theorem 3.1 and Theorem 3.3 several decision problems, such as: the *emptiness* or the *finiteness* property of the generated language, the *boundedness* or the *reachability* property of a component during the language generation process, can be solved through a structural analysis of the above trees. We recall that the *emtiness/finiteness* problem for grammar systems consists in deciding *whether the language generated by a particular grammar system is empty/finite or not.* The next theorems deal with these problems.

Theorem 3.4. *It is decidable whether the language generated by a CD grammar system, starting with an arbitrary axiom, working in* $\{\leq 1, = 1\}$*-comp.-mode is empty or not.*

Proof. In order to check whether the language generated in the $\{\leq 1, = 1\}$-comp.-mode, is empty or not we have to check whether or not there exists at least one leaf in the coverability tree labeled by a null vector or by a vector in $\mathbf{N}^m_{\omega_t}$. □

Theorem 3.5. *It is decidable whether a component of a CD grammar system, working in* $\{\leq 1, = 1\}$*-comp.-mode, with an arbitrary axiom, is ever active (or reachable) in any derivation (terminal or not). It is decidable whether a certain production of a certain component of a CD grammar system, working in* $\{\leq 1, = 1\}$*-comp.-mode, with an arbitrary axiom, is ever active (or reachable) in any (terminal) derivation.*

Proof. Let $\Gamma = (N, T, \alpha, P_1, ..., P_s)$ be a CD grammar system with s components, $s \geq 1$, and P_i be the component we are interested in. An inspection over all elements $\vartheta \in \mathcal{L}$ that label edges in $\mathcal{E}$, where $\mathcal{B}(\Gamma) = (\mathcal{V}, \mathcal{E})$ is the coverebility tree associated with Γ, such that $\vartheta(1) = i$, solves the first claim of the theorem. If we consider $\{r_1, r_2, ..., r_{|P_i|}\}$ the ordered set of rules from P_i, where $|P_i|$ is the number of rules in P_i, and r_j, $1 \leq j \leq |P_i|$, represents the rule we are interested in, then an inspection over all elements $\vartheta \in \mathcal{L}$ that label edges in $\mathcal{E}$, such that $\vartheta(1) = i$ and $\vartheta(2) = j$, solves the second claim of the theorem. □

Theorem 3.6. *It is decidable whether a component of a CD grammar system, working in the* $\{\leq 1, = 1\}$*-comp.-mode, with an arbitrary axiom, is activated of an unbounded number of times in any (terminal) derivation.*

Proof. Let $\Gamma = (N, T, \alpha, P_1, ..., P_s)$ be a CD grammar system with s components, $s \geq 1$. Let P_i be the component we are interested in. From the definition of the coverability tree, between each two consecutive nodes v and v' labeled by V and V' (V, $V' \in \mathbf{N}^m_{\omega,\omega_t}$) such that ϑ is the element in $\mathcal{L}$, enabled at vector V, we have the next firing rule $V[\vartheta\rangle_\Gamma V'$. If $\vartheta \in \mathcal{L}$ such that $\vartheta(1) = i$, and ϑ yields a vector V' that labels a cutting point in the coverability tree, then P_i is activated of an unbounded number of times. Otherwise, for any path from the root to a leaf labeled by a null vector or ω_t-vector, we check whether there exists at least one element $\vartheta \in \mathcal{L}$, such that $\vartheta(1) = i$, that yields an ω-vector, or not. □

Corollary 3.7. *It is decidable whether a component of a CD grammar system, working in the* $\{\leq 1, = 1\}$*-comp.-mode, is activated at least* q *times,* $q \geq 1$*, during a (terminal) derivation.*

Definition 3.8. Let Γ be a CD grammar system working in $\{\leq 1, = 1\}$-comp.-mode, $\mathcal{B}(\Gamma)$ and $\mathcal{B}_t(\Gamma)$ be the coverability tree and the twin coverability tree associated with Γ, respectively. We say that the place $V(q)$, $V \in \mathbf{N}^m_{\omega,\omega_t}$ (or $V \in \mathbf{N}^p_\omega$) is *bounded* in $\mathcal{B}(\Gamma)$ (or in $\mathcal{B}_t(\Gamma)$) if there exists a constant c such that $V(q) \leq c$, for any vector V that is a label of a node in $\mathcal{V}$ (or in $\mathcal{V}_t$). We say that $\mathcal{B}(\Gamma)$ (or $\mathcal{B}_t(\Gamma)$) is a *bounded coverability tree* if each place $V(q)$ is bounded, for any q, $1 \leq q \leq m$ (or $1 \leq q \leq p$), and any vector V that is a label of a node in $\mathcal{V}$ (or in $\mathcal{V}_t$).

Note that if the coverability tree associated with a CD grammar system is bounded this does not emerge that the twin coverability tree associated with it is bounded, too. Example 2.3, with the axiom $\alpha = AD$ deals with this case. That is why the boundedness property of the coverability tree is not a sufficient condition for the finiteness of the language generated by a CD grammar system. In order to cover this lack of the coverability tree analysis method we have introduced the twin coverability tree structure.

Theorem 3.9. *It is decidable whether the language generated by a CD grammar system working in the* $\{\leq 1, = 1\}$*-comp.-mode, is finite or not.*

Proof. In order to decide whether the language generated by a CD grammar system working in the $\{\leq 1, = 1\}$-comp.-mode is finite or not it is enough to check whether the twin coverability tree associated with the coverability tree of a CD grammar system is bounded or not. This is done through a searching for the ω symbol in the twin coverability tree. If no ω symbol occurs in any of the vectors that label nodes in this tree then the language is finite. It is infinite otherwise. □

4 Consequences on decision problems for grammars with regulated rewriting

Almost all decision problems for grammars with regulated rewriting have been proved to be decidable and NP-hard, see [10]. Several problems are undecidable, such as the context-freeness problem of the languages generated by λ-free matrix grammars without appearance checking or the emptiness and finiteness problems for λ-free matrix grammars with appearance checking, for instance. However, there are many other decision problems for grammars with regulated rewriting that are still open. We recall here only a few of them: the membership, emptiness and finiteness problems for ordered (O) grammars and forbidding random context (fRC) grammars, or the emptiness and finiteness problems for λ-free ordered grammars. For CD grammar systems that work in $\{\leq 1, = 1\}$-comp.-mode, from [2] we have.

Theorem 4.1. *For* $X \in \{CF, CF - \lambda\}$, $\mathcal{L}$*(fRC,X)*$\subseteq$$\mathcal{L}$*(CD, X, f-comp.),* $f \in \{\leq 1, = 1\}$.

In Section 3 we proved that the emptiness and finiteness problems are decidable for CD grammar systems working in $\{\leq 1, = 1\}$-comp.-mode. With an affirmative answer for these systems, due to the constructive proof in [2] of Theorem 4.1, we have a positive answer for the emptiness and finiteness problems for the class of languages generated by fRC grammars,

with or without λ-rules. On the other hand from [10] we know that the class of languages generated by fRC grammars equals the class of languages generated by ordered grammars, see Theorem 4.2.

Theorem 4.2. *For $X \in \{CF, CF-\lambda\}$, $\mathcal{L}$(fRC,X) = $\mathcal{L}(O, X)$.*

Due to the above theorem and due to the constructive character of the proof in [10], the emptiness and finiteness problems are decidable for ordered grammars, too. So that the following theorem holds.

Theorem 4.3. *The emptiness and finiteness problems are decidable for forbidding random context grammars and ordered grammars.*

On the other hand, it is well known from [10], that the class of languages generated by ordered grammars is closed under intersection with regular sets. Considering an arbitrary word w, for which we want to decide whether or not $w \in L$, $L \in \mathcal{L}$(O, X), we have to compute the intersection $L \cap \{w\} \in \mathcal{L}$(O,X), $X \in \{CF, CF-\lambda\}$, and decide with respect to Theorem 4.3, whether $L \cap \{w\}$ is empty or not. Thus $w \in L$ if $L \cap \{w\} \neq \emptyset$, and $w \notin L$, otherwise. From the above explanation and Theorem 4.2, we have.

Theorem 4.4. *The membership problem is decidable for forbidding random context grammars and ordered grammars.*

In this way we settled the emptiness, finiteness and membership problems for forbidding random context grammars and ordered grammars left open in [10].

5 Conclusions

In this paper we proposed two coverability structures, the *coverability tree* and the *twin coverability tree*, associated to the derivations in a CD grammar system that works in $\{\leq 1, = 1\}$-comp.-mode. A structural analysis upon these coverability trees gives positive answers to several decision problems, such as the *finiteness, emptiness, boundedness* and *reachability* problems for CD grammar systems working in $\{\leq 1, = 1\}$-comp.-mode. Using the generative power of these systems we solved several open problems such as the *finiteness, emptiness* and *membership* problems for the class of languages generated by *forbidding random context grammars* and *ordered grammars*. The structure of the coverability trees and the analysis of these trees, for the general case $\{\leq k, = k, \geq k\}$ of competence-based protocols, $k \geq 2$ are left open for further research. Our intuition is that the transition from an ω-Parikh vector to an ω_t-Parikh vector, used for the case of $\{\leq 1, = 1\}$-comp.-mode, is not possible in all the cases. This requires another method to be applied, such as the simulation of these systems by several other classes of Petri nets, described in [4], [11], [13], [15], [22].

6 Acknowledgments

The author is indebted to Professor Erzsébet Csuhaj-Varjú for the useful discussions upon the topic of this paper, and the referees for their comments concerning this work. This work has been also supported by the Agencia Espanõla de Cooperación Internacional through a Becas-MAE fellowship for 2004-2005.

References

[1] M.H. ter Beek, E. Csuhaj-Varjú, M. Holzer, G. Vaszil. On competence in cooperating distributed grammar systems, *Technical Report 2002/1*, Research Group on Modeling Multi-Agent Systems, MTA SZTAKI, Budapest. 2002.

[2] M.H. ter Beek, E. Csuhaj-Varjú, M. Holzer, G. Vaszil. On competence in CD grammar systems, *Proceedings of the 8th International DLT Conference*, Auckland, New Zealand, December 2004, LNCS 3340, 76–88. 2004.

[3] H. Bordihn, E. Csuhaj-Varjú. On competence and completeness in CD grammar systems, *Acta Cybernetica*, 12(4), 347–361. 1996.

[4] S. Crespi-Reghizzi, D. Mandrioli. Petri nets and Szilard languages, *Information and Control*, 33, 177–192. 1977.

[5] E. Csuhaj-Varjú, J. Dassow. On cooperating/distributed grammar systems, *Journal of Information Processing and Cybernetics EIK*, 26(1-2), 49–63. 1990.

[6] E. Csuhaj-Varjú, J. Dassow, M. Holzer. On competence-based cooperation strategy in CD grammar systems, *Technical Report*, 2004/3, Theoretical Computer Science Research Group, MTA SZTAKI, Budapest. 2004.

[7] E. Csuhaj-Varjú, J. Kelemen. Cooperating grammar systems: A syntactical framework for blackboard model of problem solving, *Proceedings of AIICSR'89*, I. Plander ed., Elsevier Publishing Company, Amsterdam, 121–127. 1989.

[8] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Păun. Grammar Systems: A Grammatical Approach to Distribution and Cooperation, Gordon and Breach Science Publishers, Yverdon. 1994.

[9] J. Dassow. On cooperating distributed grammar systems with competence based start and stop conditions, *Proceedings of Grammar System Week*, E. Csuhaj-Varjú and G. Vaszil eds., MTA SZTAKI, Budapest, Hungary. 2004.

[10] J. Dassow, G. Păun. Regulated Rewriting in Formal Language Theory, EATCS Monographs on Theoretical Computer Science, Springer-Verlag Berlin. 1989.

[11] C. Dufourd, A. Finkel, P. Schnoebelen. Reset nets between decidability and undecidability, *Proceedings ICALP '98*, LNCS 1443, 103–115, Springer-Verlag. 1998.

[12] Handbook of Formal Languages. Volume 1-3, G. Rozenberg and A. Salomaa eds., Springer-Verlag, Berlin. 1997.

[13] D. Hauschildt, M. Jantzen. Petri net algorithms in the theory of matrix grammars, *Acta Informatica*, Vol. 31, No. 9, 719–728. 1994.

[14] R.M. Karp, R.E. Miller. Parallel program schemata, *Journal of Computer and System Science*, 3, 147–195. 1969.

[15] R. Melinte, O. Oanea, I. Olga, F.L. Ţiplea. The home marking problem and some related concepts, *Acta Cybernetica*, 15(3), 467–478. 2002.

[16] V. Mihalache. On the expressiveness of coverability trees for PC grammar systems, *In Grammatical Models of Multi-Agent Systems*, G. Păun, A. Salomaa eds., Gordon and Breach Science Publishers, 86–98. 1999.

[17] V. Mihalache. Cooperation, Communication, Control. Investigations on Grammar Systems, *TUCS Dissertations*, No 9, June. 1998.

[18] V. Mihalache. Decidability problems in grammar systems, *Theoretical Computer Science*, 215, 169–189. 1999.

[19] W. Reisig. Petri Nets. An Introduction, EATCS Monographs on Theoretical Computer Science, Springer-Verlag Berlin. 1985.

[20] F.L. Ţiplea, C. Ene. A Coverability structure for parallel communicating grammar systems, *Journal Information Processing Cybernetics, EIK*, 29, (5), 303–315. 1993.

[21] F.L. Ţiplea, C. Ene, C.M. Ionesco, O. Procopiuc. Some decision problems for parallel communicating grammar systems, *Theoretical Computer Science*, 134, 365–385. 1994.

[22] F.L. Ţiplea. On conditional grammars and conditional Petri nets, *In Mathematical Aspects of Natural and Formal Languages*, Gh.Păun ed., World Scientific, Singapore, 431–456. 1994.

Liliana Cojocaru
Research Group on Mathematical Linguistics
Rovira i Virgili University of Tarragona
Pl. Imperial Tarraco 1, 43005, Spain
liliana.cojocaru@estudiants.urv.es

Adaptive and Hybrid Algorithms: classification and illustration on triangular system solving*

Van Dat Cung Vincent Danjean Jean-Guillaume Dumas
Thierry Gautier Guillaume Huard Bruno Raffin Christophe Rapine
Jean-Louis Roch Denis Trystram

Abstract

We propose in this article a classification of the different notions of hybridization and a generic framework for the automatic hybridization of algorithms. Then, we detail the results of this generic framework on the example of the parallel solution of multiple linear systems.

Introduction

Large-scale applications, software systems and applications are getting increasingly complex. To deal with this complexity, those systems must manage themselves in accordance with high-level guidance from humans. Adaptive and hybrid algorithms enable this self-management of resources and structured inputs. In this paper, we propose a classification of the different notions of hybridization and a generic framework for the automatic hybridization of algorithms. We illustrate our framework in the context of combinatorial optimizations and linear algebra, in a sequential environment as well as in an heterogeneous parallel one. In the sequel, we focus on hybrid algorithms with provable performance. Performance is measured in terms of sequential time, parallel time or precision.
After surveying, classifying and illustrating the different notions of hybrid algorithms in section 1, we propose a generic recursive framework enabling the automation of the process of hybridization in section 2. We then detail the process and the result of our generic hybridization on the example of solving linear systems in section 3.

1 A survey and classification of hybrid algorithms

1.1 Definitions and classification

In this section we propose a definition of *hybrid* algorithm, based on the notion of strategic choices among several algorithms. We then refine this definition to propose a classification

*This work is supported by the INRIA-IMAG project AHA: Adaptive and Hybrid Algorithms.

of hybrid algorithms according to the number of choices performed (*simple, baroque*) and the amount of inputs/architecture information used (*tuned, adaptive, introspective, oblivious, engineered*). Figure 1 summarizes this classification.

Definition 1.1 (Hybrid). An algorithm is *hybrid* (or a poly-algorithm) when there is a choice at a high level between at least two distinct algorithms, each of which could solve the same problem.

The choice is strategic, not tactical. It is motivated by an increase of the performance of the execution, depending on both input/output data and computing resources. The following criterion on the number of choices to decide is used to make a first distinction among *hybrid* algorithms.

Definition 1.2 (Simple versus Baroque). A *hybrid* algorithm may be

- *simple*: $O(1)$ choices are performed whatever the input (e.g. its size) is. Notice that, while only a constant number of choises are done, each choice can be used several times (an unbounded number of times) during the execution. Parallel divide&conquer algorithms illustrate this point in next section.
- *baroque*: the number of choices is not bounded: it depends on the input (e.g. its size).

While choices in a simple *hybrid* alogrithm may be defined statically before any execution, some choices in *baroque hybrid* algorithms are necessarily computed at run time.
The choices may be performed based on machine parameters. But there exist efficient algorithms that do not base their choices on such parameters. For instance, cache-oblivious algorithms have been successfully explored in the context of regular [11] and irregular [1] problems, on sequential and parallel machine models [2]. They do not use any information about memory access times, or cache-line or disk-block sizes. This motivates a second distinction based on the information used.

Definition 1.3 (oblivious, tuned, engineered, adaptive, introspective). Considering the way choices are computed, we distinguishe the following class of *hybrid* algorithms:

- A *hybrid* algorithm is *oblivious*, if its control flow depends neither on the particular values of the inputs nor on static properties of the resources.
- A *hybrid* algorithm is *tuned*, if a strategic decision is made based on static resources such as memory specific parameters or heterogeneous features of processors in a distributed computation.
  A *tuned* algorithm is *engineered* if a strategic choice is inserted based on a mix of the analysis and knowledge of the target machine and input patterns. A *hybrid* algorithm is *self-tuned* if the choices are automatically computed by an algorithm.
- A *hybrid* algorithm is *adaptive* if it avoids any machine or memory-specific parameterization. Strategic decisions are made based on resource availability or input data

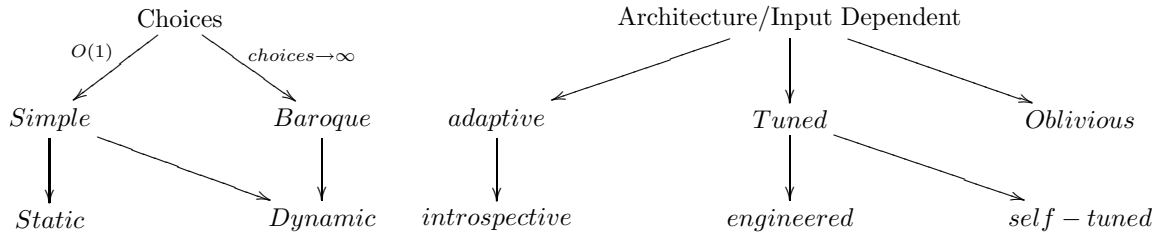


Figure 1: Classification of *hybrid* algorithms

properties, both discovered at run-time (such as idle processors).
An adaptive algorithm is *introspective* if a strategic decision is made based on assessment of the algorithm performance on the given input up to the decision point.

In [12], Ganek and Corbi defined *autonomic computing* to be the conjunction of self-configuring, self-healing, self-optimizing and self-protecting systems. Self-configuring relates to what we call adaptivity, self-optimizing to self-tuning. Autonomic computing thus adds fault-tolerance (self-healing) and security (self-protecting) to our notion of *hybrid* computing. Above definitions deliberately focus on a general characterization of adaptation in the algorithm. They consider neither implementation nor performance. To implement an *adaptive* algorithm, we may distinguish two approaches. Either the choices are included in the algorithm itself, or they may be inserted dynamically to change the software itself, or its execution environment. An algorithm is *evolutive* (or interactive) if a strategic choice is inserted dynamically. Reflexive languages enable to change the behavior of a program dynamically [18]. Polymorphism or template specialization is a way to optimize an algorithm. We view polymorphism and template mechanisms as a possible way to implement the different kinds of *hybrid* algorithm we propose.

1.2 Illustrations on examples

We illustrate the previous criteria on some examples of *hybrid* algorithms or libraries.

BLAS libraries. ATLAS [23] and GOTO [**?**] are libraries that implement basic linear algebra subroutines. Computation on matrices are performed by blocks. The block size and the sequential algorithm used for a basic block are chosen based on performance measures on the target architecture. The decisions are computed automatically at installation with ATLAS while they are provided only for some architectures with GOTO. ATLAS implements *self-tuned simple hybrid* algorithms and GOTO *simple engineered* ones.

Granularity in sequential divide&conquer algorithms. Halting recursion in divide&conquer to complete small size computations with another more efficient algorithm is

a classical technique to improve the practical performance of sequential algorithms. The resulting algorithm is a *simple hybrid* one. Often the recursion threshold is based on resource properties. This is the case for the GMP integer multiplication algorithm that successively couples four algorithms: Schönhage-Strassen $\Theta(n \log n \log\log n)$, Toom-Cook 3-way ($\Theta(n^{1.465})$), Karatsuba $\Theta(n^{log_2 3})$ and standard $\Theta(n^2)$ algorithms.

Linpack benchmark for parallel LU factorization. Linpack [6] is one milestone in parallel machines' power evaluation. It is the reference benchmark for the top-500 ranking of the most powerful machines. The computation consists in a LU factorization, with raw partial pivoting in the "right-looking" variant [6], the processors assumed being identical. To limit the volume of communication to $O(n^2\sqrt{p})$, a cyclic bidimensional block partitioning is used on a virtual grid of $q^2 = p$ processors. The block (i, j) is mapped to the processor of index $P(i, j) = (i \bmod q)q+(j \bmod q)$ and operations that modify block (i, j) are scheduled on processor $P(i, j)$. Linpack has a standard implementation on top of MPI with various parameters that may be tuned: broadcast algorithm (for pivot broadcasting on a line of processors), level of recursion in the "right-looking" decomposition algorithm and block size. The parallel architecture may also be tuned to improve the performance [21]. Linpack is an *engineered tuned simple hybrid* algorithm.

FFTW. FFTW [10] is a library that implements discrete Fourier transform of a vector of size n. We summarize here the basic principle of FFTW. For all $2 \leq q \leq \sqrt{n}$, the FFT Cooley-Tuckey recursive algorithm reduces to q FFT subcomputations of size $\left\lceil \frac{n}{q} \right\rceil$ and $\left\lceil \frac{n}{q} \right\rceil$ FFT subcomputations of size q, plus $O(n)$ additional operations. Hybridization in FFTW occurs at two levels:

- at installation on the architecture. For a given n_0 the best unrolled FFT algorithm for all $n \leq n_0$ is chosen among a set of algorithms by experimental performance measurements. This *hybrid* algorithm is *simple tuned*.
- at execution. FOr a given size n of the input vectors and for all $n_0 \leq m \leq n$, a planner precomputes the splitting factor q_m that will be further used for any recursive FFT with size m. This precomputation is performed by dynamic programming: it optimizes each sub-problem of size m locally, independently of the larger context where it is invoked. The planner adds a precomputation overhead. This overhead may be amortized by using the same plan for computing several FFTs of the same size n. FFTW3 also proposes heuristic algorithms to compute plans with smaller overhead than dynamic programming.

The number of choices in FFTW depends on the size n of the inputs. FFTW is a *self-tuned baroque hybrid* algorithm.

Granularity in parallel divide&conquer algorithms. Parallel algorithms are often based on a mix of two algorithms: a sequential one that minimizes the number of operations

T_1 and a parallel one that minimizes the parallel time T_∞. The cascading divide&conquer technique [16] is used to construct a *hybrid* algorithm with parallel time $O(T_\infty)$ while performing $O(T_1)$ operations. For instance, iterated product of n elements can be performed in parallel time $T_\infty = 2.\log n$ with $\frac{n}{\log n}$ processors by choosing a grain size of $\log n$. Even if this choice depends on the input size n, it can be computed only once at the beginning of the execution. The algorithm is a *simple hybrid* one.
Other examples of such parallel *simple hybrid* algorithms are: computation of the maximum of n elements in asymptotic optimal time $\Theta(\log\log n)$ on a CRCW PRAM with $\frac{n}{\log\log n}$ processors [16]; solving of a triangular linear system in parallel time $O(\sqrt{n}\log n)$ with $\Theta(n^2)$ operations [19]. In section 3 we detail an extended *baroque* hybridization for this problem, enabling a higher performance on a generic architecture.

Parallel *adaptive* algorithms by work-stealing - Kaapi. Cilk [17], Athapascan/Kaapi [15] and Satin [22] are parallel programming interfaces that support recursive parallelism and implement a work-stealing scheduling based on the *work first principle*. A program explicits parallelism and synchronization. While Cilk and Satin are restricted to serie-parallel tasks DAGs, Kaapi accepts any kind of dataflow dependencies. However, all are based on a sequential semantics: both depth first sequential search (DFS) and width (or breadth) first parallel search (BFS) are correct executions of the program. Then the program implements a parallel algorithm (BFS) that can also be considered as a sequential one (DFS). The (recursive) choices between both are performed by the scheduler. To save memory, depth-first execution (DFS) is always locally preferred. When a processor becomes idle, it steals the oldest ready task on a non-idle processor This stealing operation then corresponds to a breadth first execution (BFS). Since each parallel task creation can be performed either by a sequential call (DFS algorithm) or by creation of a new thread (BFS algorithm) depending on resource idleness, any parallel program with non-fixed degree of parallelism is a *hybrid baroque* algorithm. Because the choice does not depend on the input size but only on resource idleness, the algorithm is *adaptive*. In section 2.3 we detail a more general coupling for this problem.

2 Generic algorithmic schemes for *hybrid* algorithms

In this section we detail a generic scheme to control the time overhead due to choices in a *hybryd* algorithm, providing a proven upperbound for sequential and parallel *baroque* hybridization.

2.1 Basic representation

Let f be a problem with input set I and output set O. For the computation of f, a *hybrid* algorithm is based on the composition of distinct algorithms $(f_i)_{i=1,\ldots,k}$, each solving the problem f. Since an algorithm is finite, the number $k \geq 2$ of algorithms is finite; however, each of those algorithms may use additional parameters, based on the inputs, outputs or machine parameters (e.g. number of processors).

We assume that each of those algorithms is written in a recursive way: to solve a given instance of f, algorithm f_i reduces it to subcomputations instances of f of smaller sizes. Hybridization then consists in choosing for each of those subcomputations the suited algorithm f_j to be used (fig. 2). This choice can be implemented in various ways. For instance, f may

```
Algorithm f_i ( I n, input, O output, ... ) {
  ...
  f( n-1, ... ) ;
  ...
  f( n / 2, ... ) ;
  ...
};
```

Figure 2: Recursive description of a *hybrid* algorithm f_i.

be implemented as a pure virtual function, each of the f_i being an inherited specialization.

Scheme for decreasing overhead due to choices. For *baroque* algorithms the choices between the different f_i's are performed at runtime. Therefore an important problem is related to reducing the overhead related to the computation of each choice. In the next section, we describe an original alternative scheme to decrease the overhead induced by the choices for each call to f in the previous algorithm. Generalization to various computations [5] (namely Branch&X computations and linear algebra) is based on the use of an exception mechanism. For a given subcomputation, a default given computation f_j is favored. However, this choice may be changed under some exceptional circumstances depending on values or machine parameters. Then, if the total number of such exceptions is small with respect to the total number of subcomputations, the overhead due to choices become negligible. We detail such a scheme in next section.

2.2 *Baroque* coupling of sequential and parallel algorithms

We presented the coupling of a sequential algorithm f_{seq} and a parallel one f_{par} that solve the same problem f. For the sake of simplicity, we assume that the sequential algorithm performs a first part of the sequential computation (called *ExtractSeq*) and then performs a recursive terminal call to f to complete the computation. Besides, we assume that the sequential algorithm is such that at any time of its execution, the sequence of operations that completes the algorithm f_{seq} can be performed by another parallel recursive (fine grain) algorithm f_{par}. The operation that consists in extracting the last part of the sequential computation in progress to perform it in parallel with f_{par} is called *ExtractPar*. After completion of f_{par}, the final result is computed by merging both the result of the first part computed by f_{seq} (not affected by *ExtractPar*) and the result of the *ExtractPar* part computed by f_{par}.

More precisely, given a sequential algorithm f_{seq} (resp. parallel f_{par}), the result r of its evaluation on an input x is denoted $f_{\text{seq}}(x)$ (resp. $f_{\text{par}}(x)$). We assume that x has a list structure with a concatenation operator $\sharp$ and that there exists an operator $\oplus$ (not necessarily associative) for merging the results. At any time of evaluation of $f_{\text{seq}}(x)$, x can be split into $x_1 \sharp x_2$, due to either an *ExtractSeq* or an *ExtractPar* operation on x. The result computed by the parallel algorithm is then $f_{\text{par}}(x) = f(x_1) \oplus f(x_2)$. We assume that both results $f_{\text{seq}}(x)$ and $f_{\text{par}}(x)$ are equivalents with respect to a given measure. In the restricted framework of list homomorphism [3], this hypothesis can be written as $f(x \sharp y) = f(x) \oplus f(y)$. However, it is possible to provide parallel algorithms for problems that are not list homomorphisms [4] at the price of an increase in the number of operations.
To decrease overhead related to choices for f between f_{seq} and f_{par}, f_{seq} is the default choice used. Based on a workstealing scheduling, f_{par} is only chosen when a processor becomes idle, which leads to an *ExtractPar* operation.
This exception mechanism can be implemented by maintaining during any execution of $f_{\text{seq}}(x)$ a lower priority process ready to perform an *ExtractPar* operation on x resulting in an input x_2 for f_{par} only when a processor becomes idle.
Then the overhead due to choices is only related to the number of *ExtractPar* operations actually performed.
To analyze this number, we adopt the simplified model of Cilk-5 [17] also valid for *Kaapi* [15]. It relies on Graham's bound (see Equation 2 in [17]). Let $T_1^{(\text{seq})}$ (resp. $T_1^{(\text{par})}$) be the execution time on a sequential processor (i.e. work) of f_{seq} (resp. f_{par}), and let $T_\infty^{(\text{par})}$ be the execution time of f_{par} on an unbounded number of identical processors.

Theorem 2.1. *When the* hybrid *program is executed on a machine with m identical processors, the number of choices that result in a choice f_{par} for f instead of f_{seq} is bounded by $(m-1).T_\infty^{(\text{par})}$*

Proof. On an infinite number of processors, all the computation is performed by f_{par}; the parallel time of the *hybrid* algorithm is then $T_\infty^{(\text{par})}$. Then the number of steal requests is bounded by $T_\infty^{(\text{par})}$ on each processor (Graham's bound), except for the one running f_{seq}. The latter only executes the sequential algorithm, but is subject to ExtractPar, due to steal requests from the others. This is true for any execution of such *hybrid baroque* algorithm. □

The consequence of this theorem is that for a fine grain parallel algorithm that satisfies $T_\infty^{(\text{par})} \ll T_1^{(\text{seq})}$, even if the *hybrid* algorithm is *baroque* (non constant number of choices), the overhead in time due to choices in the *hybrid* algorithm is negligible when compared to the overall work.

Remark. The overhead due to the default call to *ExtractSeq* can also be reduced. Ideally, *ExtractSeq* should extract a data whose computations by f_{par} would require a time at least $T_\infty^{(\text{par})}$, which is the critical time for f_{par}.

2.3 Application to the coupling of DFS/BFS for combinatorial optimization

The performance and overhead of the previous scheme were experimentally determined for the *Quadratic Assignment Problem* (for instance NUGENT 22[1]). This application implements a Branch&Bound algorithm: it recursively generates nodes in the search tree, which has 221938 nodes and a maximal depth of 22.
Locally, each processor implements by default a sequential algorithm f_{seq} that implements a depth first search (DFS) in the tree. It enables to save memory and also to optimize branching in the tree without copy (sons of a node n are sequentially created from the value of n with backtracking). To minimize critical time, the alternative f_{par} parallel algorithm implements a breadth first search (BFS) algorithm. When a processor becomes idle, it picks the oldest node of a randomly chosen non-idle processor (*ExtractPar*). This parallel algorithm introduces an overhead due to node copy.
The experiments were conducted on the iCluster2[2], a cluster of 104 nodes interconnected by a 100Mbps Ethernet network. Each node features two Itanium-2 processors (900 MHz) and 3 GB of local memory. The algorithm was parallelized using Kaapi. The degree of parallelism (threshold) can be adjusted: after a given depth, the subtree of a node is computed locally by f_{seq}. This thresold defined the minimum granularity and should be chosen such that the time of the local computation by f_{seq} is comparable to the time overhead of parallelism.

Figure 3: Impact of granularity

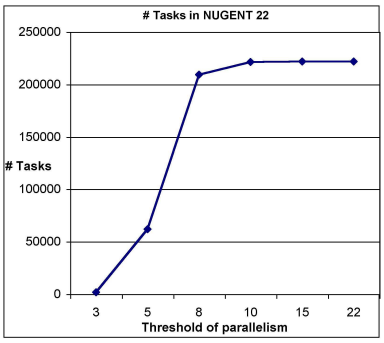
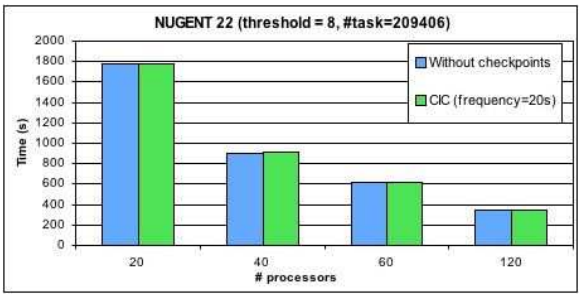


Figure 4: Execution time (sequential time: 34,695s)



The sequential execution time (C++ code without Kaapi) was 34,695 seconds. With Kaapi, at fine grain (threshold ≥ 10), the execution on a single processor generated 225,195 tasks and ran in 34,845 seconds. The impact of the degree of parallelism can be seen in Figure 3 that gives the number of parallel tasks generated for different thresholds. The degree of parallelism increases drastically for threshold 5 and approaches its maximum at threshold 10. Figure 4 shows that the application is scalable with a fine threshold (8, i.e. 209406 nodes). Since the critical time T_∞ is small, there are few successful steals and the overhead of hybridation between f_{seq} and f_{par} has small impact on efficiency.

[1]http://www.opt.math.tu-graz.ac.at/qaplib
[2]http://www.inrialpes.fr/sed/i-cluster2

Notice that Kaapi also includes a (*hybrid*) checkpoint/restart mechanism [15] to support the resilience and the addition of processors. This features makes the application itself *oblivious* to dynamic platforms. The overhead of this checkpoint mechanism appears negligible for this application (Figure 4).
In the next section, we detail various forms of hybridation on a single example, the solving of a triangular system.

3 Hybridization for triangular system solving

3.1 Triangular system solving with matrix right-hand side

Exact matrix multiplication, together with matrix factorizations, over finite fields can now be performed at the speed of the highly optimized numerical BLAS routines. This has been established by the FFLAS and FFPACK libraries [8, 9]. In this section we discuss the implementation of exact solvers for triangular systems with matrix right-hand side (or equivalently left-hand side). This is also the simultaneous resolution of n triangular systems. Without loss of generality for the triangularization, we here consider only the case where the row dimension, m, of the the triangular system is less than or equal to the column dimension, n. The resolution of such systems is e.g. the main operation in block Gaussian elimination. For solving triangular systems over finite fields, the block algorithm reduces to matrix multiplication and achieves the best known algebraic complexity. Therefore, from now on we will denote by ω the exponent of square matrix multiplication (e.g. from 3 for classical, to 2.375477 for Coppersmith-Winograd). Moreover, we can bound the arithmetical cost of a $m \times k$ by $k \times n$ rectangular matrix multiplication (denoted by $R(m,k,n)$) as follows: $R(m,k,n) \leq C_\omega min(m,k,n)^{\omega-2} max(mk,mn,kn)$ [14]. In the following subsections, we present the block recursive algorithm and two optimized implementation variants.

3.2 Scheme of the block recursive algorithm

The classical idea is to use the divide and conquer approach. Here, we consider the upper left triangular case without loss of generality, since any combination of upper/lower and left/right triangular cases are similar: if U is upper triangular, L is lower triangular and B is rectangular, we call `ULeft-Trsm` the resolution of $UX = B$. Suppose that we split the matrices into blocks and use the divide and conquer approach as follows:

$$\overbrace{\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}}^{X} = \overbrace{\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}}^{B}$$

1. X_2 :=`ULeft-Trsm`(A_3, B_2);
2. $B_1 := B_1 - A_2 X_2$;
3. X_1 :=`ULeft-Trsm`(A_1, B_1);

With $m = n$ and classical matrix multiplication, the arithmetic cost of this algorithm is $TRSM(m) = m^3$ as shown e.g. in [9, Lemma 3.1].
We also now give the cost of the triangular matrix multiplication, TRMM, and of the triangular inversion, INVT, as we will need them in the following sections.
To perform the multiplication of a triangular matrix by a dense matrix via a block decomposition, one requires four recursive calls and two dense matrix-matrix multiplications. The cost is thus $TRMM(m) = 4TRMM(m/2) + 2MM(m/2)$. The latter is $TRMM(m) = m^3$ with classical matrix multiplication.
Now the inverse of a triangular matrix requires two recursive calls to invert A_1 and A_3. Then, the square block of the inverse is $-A_1^{-1}A_2A_3^{-1}$. The cost is thus $INVT(m) = 2INVT(m/2) + 2TRMM(m/2)$. The latter is $INVT(m) = \frac{1}{3}m^3$ with classical matrix multiplication.

3.3 Two distinct *hybrid* degenerations

3.3.1 Degenerating to the BLAS "dtrsm"

Matrix multiplication speed over finite fields was improved in [8, 20] by the use of the numerical BLAS[3] library: matrices were converted to floating point representations (where the linear algebra routines are fast) and converted back to a finite field representation afterwards. The computations remained exact as long as no overflow occurred. An implementation of `ULeft-Trsm` can use the same techniques. Indeed, as soon as no overflow occurs one can replace the recursive call to `ULeft-Trsm` by the numerical BLAS *dtrsm* routine. But one can remark that approximate divisions can occur. So we need to ensure both that only exact divisions are performed and that no overflow appears. However when the system is unitary (only 1's on the main diagonal) the division are of course exact and will even never be performed. Our idea is then to transform the initial system so that all the recursive calls to `ULeft-Trsm` are unitary. For a triangular system $AX = B$, it suffices to factor first the matrix A into $A = UD$, where U, D are respectively an upper unit triangular matrix and a diagonal matrix. Next the unitary system $UY = B$ is solved by any `ULeft-Trsm` (even a numerical one), without any division. The initial solution is then recovered over the finite field via $X = D^{-1}Y$. This normalization leads to an additional cost of $O(mn)$ arithmetic operations (see [9] for more details).
We now care for the coefficient growth. The use of the BLAS routine trsm is the resolution of the triangular system over the integers (stored as `double` for `dtrsm`). The restriction is the coefficient growth in the solution. Indeed, the k^{th} value in the solution vector is a linear combination of the $(n - k)$ already computed next values. This implies a linear growth in the coefficient size of the solution, with respect to the system dimension: for a given p, the dimension n of the system must satisfy $\frac{p-1}{2}\left[p^{n-1} + (p-2)^{n-1}\right] < 2^{m_a}$ where m_a is the size of the mantissa [9]. Then the resolution over the integers using the BLAS trsm routine is exact. For instance, with a 53 bits mantissa, this gives quite small matrices, namely at most 55×55 for $p = 2$, at most 4×4 for $p \leq 9739$, and at most $p = 94906249$ for 2×2 matrices.

[3] `www.netlib.org/blas`

Nevertheless, this technique is speed-worthy in many cases.
In the following, we will denote by $S_{BLAS}(p)$ the maximal matrix size for which the BLAS resolution is exact. Also, `BLASTrsm` is the recursive block algorithm, switching to the BLAS resolution as soon as the splitting gives a block size lower than $S_{BLAS}(p)$.

3.3.2 Degenerating to delayed modulus

In the previous section we noticed that BLAS routines within `Trsm` are used only for small systems. An alternative is to change the cascade: instead of calling the BLAS, one could switch to the classical iterative algorithm: Let $A \in {\mathbb{Z}_{/p\mathbb{Z}}}^{m\times m}$ and $B, X \in {\mathbb{Z}_{/p\mathbb{Z}}}^{m\times n}$ such that $AX = B$, then $\forall i, X_{i,*} = \frac{1}{A_{i,i}}(B_{i,*} - A_{i,[i+1..m]}X_{[i+1..m],*})$ The idea is that the iterative algorithm computes only one row of the whole solution at a time. Therefore its threshold t is greater than the one of the BLAS routine, namely it requires only $t(p-1)^2 < 2^{m_a}$ for a $0..p-1$ unsigned representation, or $t(p-1)^2 < 2^{m_a+1}$ for a $\frac{1-p}{2}..\frac{p-1}{2}$ signed one. Now we focus on the dot product operation, base for matrix-vector product. According to [7], where different implementations of a dot product are proposed and compared on different architecture (Zech log, Montgomery, float, ...), the best implementation is a combination of a conversion to floating point representation with delayed modulus (for big prime and vector size) and an overflow detection trick (for smaller prime and vector size).
$\texttt{DelayTrsm}_t$ is the recursive block algorithm, switching to the delayed iterative resolution as soon as the splitting gives a block size lower than t (of course, t must satisfy $t \leq S_{BLAS}(p)$).

3.4 Tuning the "Trsm" implementation

3.4.1 Experimental tuning

As shown in section 3.2 the block recursive algorithm `Trsm` is based on matrix multiplications. This allows us to use the fast matrix multiplication routine of the FFLAS package [8]. This is an exact wrapping of the ATLAS library[4] used as a kernel to implement the `Trsm` variants. The following table results from experimental results of [9] and expresses which of the two preceding variants is better. `Mod<double>` is a field representation from [7] where the elements are stored as floating points to avoid one of the conversions. `G-Zpz` is a field representation from [13] where the elements are stored as small integers.

n	*400*	*700*	*1000*	*2000*	*5000*
Mod<double>(5)	BLASTrsm	BLASTrsm	BLASTrsm	BLASTrsm	BLASTrsm
Mod<double>(32749)	$\texttt{DelayTrsm}_{50}$	$\texttt{DelayTrsm}_{50}$	$\texttt{DelayTrsm}_{50}$	BLASTrsm	BLASTrsm
G-Zpz(5)	$\texttt{DelayTrsm}_{100}$	$\texttt{DelayTrsm}_{150}$	$\texttt{DelayTrsm}_{100}$	BLASTrsm	BLASTrsm
G-Zpz(32749)	$\texttt{DelayTrsm}_{50}$	$\texttt{DelayTrsm}_{50}$	$\texttt{DelayTrsm}_{50}$	$\texttt{DelayTrsm}_{50}$	$\texttt{DelayTrsm}_{50}$

Table 1: Best variant for `Trsm` on a P4, 2.4GHz

In the following, we will denote by $S_{Del}(n,p)$ the threshold t for which $\texttt{DelayTrsm}_t$ is the most efficient routine for matrices of size n. $S_{Del}(n,p)$ is set to 0 if e.g. the `BLASTrsm` routine

[4] `http://math-atlas.sourceforge.net`[23]

is better. The experiment shows that $S_{Del}(n,p)$ can be bigger or smaller than $S_{BLAS}(p)$ depending on the matrix size, the prime and the underlying arithmetic implementation.

3.4.2 *Hybrid tuned* algorithm

The experimental results of previous section, thus provide us with an *hybrid* algorithm where we can tune some static threshold in order to benefit from all the variants. Moreover, some choices have to be made for the splitting size k in order to reach the optimal complexity T_{opt}:

$$Topt(m) = Min_k\{Topt(k) + Topt(m-k) + R(m-k,k,n)\}.$$

Algorithm `ULeft-Trsm`(A,B)
Input: $A \in {\mathbb{Z}_{/p\mathbb{Z}}}^{m\times m}$, $B \in {\mathbb{Z}_{/p\mathbb{Z}}}^{m\times n}$.
Output: $X \in {\mathbb{Z}_{/p\mathbb{Z}}}^{m\times n}$ such that $AX = B$.

if $m \leq S_{Del}(m,p)$ **then** *// Hybrid modulus degeneration 3.3.2*
  $X :=$ `DelayTrsm`(A,B);
else if $m \leq S_{BLAS}(p)$ **then** *// Hybrid BLAS degeneration 3.3.1*
  $X :=$ `BLASTrsm`(A,B);
else *// Hybrid block recursive 3.2*
  $k := Choice(1..\lfloor \frac{m}{2} \rfloor)$;
  Split matrices into k and m−k blocks $\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$
  $X_2 :=$`ULeft-Trsm`(A_3, B_2);
  $B_1 := B_1 - A_2X_2$;
  $X_1 :=$`ULeft-Trsm`(A_1, B_1);
return X;

3.5 *Baroque hybrid* parallel Trsm

The previous algorithm takes benefit of parallelism at the level of Blas matrix product operations. However, using the scheme proposed in §2.2, it is possible to obtain an algorithm with more parallelism in order to decrease the critical time when more processors are available. Furthermore, this also improves the performance of the distributed work-stealing scheduler.
Indeed, while X_2 and B_1 are being computed, additional idle processors may proceed to the parallel computation of A_1^{-1}. Indeed, X_1 may be computed in two different ways:

i. $X_1 = TRSM(A_1, B_1)$: the arithmetic cost is $T_1 = k^3$ and $T_\infty = k$;

ii. $X_1 = TRMM(A_1^{-1}, B_1)$: the arithmetic cost is the same $T_1 = k^3$ but $T_\infty = \log k$.

Indeed the version (ii) with TRMM is more efficient on a parallel point of view: the two recursive calls and the matrix multiplication in (ii) (TRMM) are independent. They can be performed on distinct processors requiring less communications than TRSM.

Since precomputation of A_1^{-1} increases the whole arithmetic cost, it is only performed if there are extra unused processors during the computation of X_2 and B_1; the latter has therefore higher priority.
The problem is to decide the size k of the matrix A_1 that will be inverted in parallel. With the optimal value of k, the computation of A_1^{-1} completes simultaneously with that of X_2 and B_1. This optimal value of k depends on many factors: number of processors, architecture of processors, subroutines, data. The algorithm presented in the next paragraph uses the *oblivious adaptive* scheme described in 2.2. to estimate this value at runtime using the *hybrid* coupling of a "sequential" algorithm f_s with a parallel one f_p.

3.5.1 Parallel *adaptive* TRSM

We assume that the parallel *hybrid* TRSM is spawned by a high priority process. Then the parallel *hybrid* TRSM consists in computing concurrently in parallel (Figure 5):

- "sequential" computation (f_s) at high priority: bottom-up computation of $X = TRSM(A, B)$ till reaching k, implemented by BUT algorithm (Bottom-Up TRSM - §A.1); all processes that perform parallel BLAS operations in BUT are executed at high priority;

- parallel computation (f_p) at low priority: parallel top-down inversion of A till reaching k, implemented by TDTI algorithm (Top Down Triangular Inversion - §A.2); all processes that participates in parallel TDTI are executed at low priority.

Algorithm `HybridParallelTrsm`$(A; B)$
Input: $A \in {\mathbb{Z}_{/p\mathbb{Z}}}^{m\times m}$, $B \in {\mathbb{Z}_{/p\mathbb{Z}}}^{m\times n}$.
Output: $X \in {\mathbb{Z}_{/p\mathbb{Z}}}^{m\times n}$ such that $AX = B$.
$k_{TDTI} := 0$; $k_{BUT} := m$;
Parallel {
At high priority: $(X_2, B_1') := BUT(A, B)$;
At low priority: $M := TDTI(\emptyset, A)$;
}
Here, BUT has stopped TDTI and $k_{BUT} \leq k_{TDTI}$.
Now, let $A_1^{'-1} = M_{1..k_{BUT},1..k_{BUT}}$;
$X_1 := A_1^{'-1}.B_1'$;

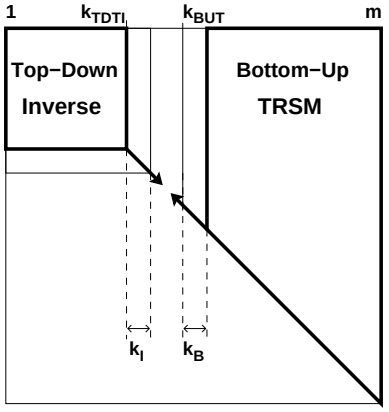


Figure 5: Parallel *adaptive* TRSM

At each step, the sequential bottom-up BUT algorithm (resp. the parallel top-down TDTI) performs an ExtractSeq (resp. ExtractPar) operation on a block of size k_B (resp. k_I) (Figure 5 and detailed subroutines BUT and TDTI in appendices). Note that the values of k_B and k_I may vary during the execution depending on the current state.

3.5.2 Definiton of parameters k_I and k_B

Parameters k_B (resp. k_I) corresponds to the *ExtractSeq* (resp. *ExtractPar*) operations presented in §2.2. The choice of their values is performed at each recursive step, depending on resources availability. This section analyzes this choice in the case where only one system is to be solved, i.e. $n = 1$.
Let $r = k_{BUT} - k_{TDTI}$.

- On the one hand, to fully exploit parallelism, k_B should not be larger than the critical time T_∞ of TDTI, i.e. $k_B = \log^2 r$.
- On the other hand, in order to keep an $O(n^2)$ number of operations if no more processors become idle, the number of operations $O(k_I^3)$ required by $TDTI$ should be balanced by the cost of the update, i.e. $k_I.r$, which leads to $k_I = \sqrt{r}$.

With those choices of k_I and k_B, and assuming that there are enough processors, the number of choices for k_I (and so k_B) will then be $O(\sqrt{r})$; the cost of the resulting *hybrid* algorithm becomes $T_1 = O(n^2)$ and $T_\infty = O(\sqrt{n}\log^2(n))$, a complexity similar to the one proposed in [19] with a fine grain parallel algorithm, while this one is coarse grain and dynamically adapts to resource idleness. Notice that if only a few processors are available, the parallel algorithm will be executed at most on one block of size $\sqrt{n}$. The BUT algorithm will behave like the previous *hybrid tuned* TRSM algorithm. Also, the algorithm is *oblivious* to the number of resources and their relative performance.

4 Conclusion

Designing efficient *hybrid* algorithms is the key to get most of the available resources and most of the structure of the inputs of numerous applications as we have shown e.g. for linear algebra or for combinatorial optimization Branch&X. In this paper, we have proposed a classification of the distinct forms of *hybrid* algorithms and a generic framework to express this adaptivity. On a single simple example, namely solving linear systems, we show that several of these "hybridities" can appear. This enables an effective hybridization of the algorithm and a nice way to adapt automatically its behavior, independent of the execution context. This is true in a parallel context where coupling of algorithms is critical to obtain a high performance.
The resulting algorithm is quite complex but can be automatically generated in our simple framework. The requirements are just to provide recursive versions of the different methods. In the AHA group[5], such coupling are studied in the context of many examples: vision and adaptive 3D-reconstruction, linear algebra in general, and combinatorial optimization.

Acknowledgments. The authors gratefully acknowledge David B. Saunders for useful discussions and suggestions for the classification of *hybrid* algorithms.

[5] aha.imag.fr

References

[1] Michael A. Bender, Erik D. Demaine, and Martin Farach-Colton. Cache-oblivious b-trees. *SIAM J. Comput.*, 35(2):341–358, 2005.

[2] Michael A. Bender, Jeremy T. Fineman, Seth Gilbert, and Bradley C. Kuszmaul. Concurrent cache-oblivious b-trees. In *SPAA'05: Proceedings of the 17th annual ACM symposium on Parallelism in algorithms and architectures*, pages 228–237, New York, NY, USA, 2005. ACM Press.

[3] R.S. Bird. *Logic of Programming and Calculi of Discrete Design*, chapter Introduction to the Theory of Lists. Springer-Verlag, 1987.

[4] M. Cole. Parallel programming with list homomorphisms. *Parallel Processing Letters*, 5(2):191–204, 1995.

[5] El-Mostafa Daoudi, Thierry Gautier, Aicha Kerfali, Rémi Revire, and Jean-Louis Roch. Algorithmes parallèles à grain adaptatif et applications. *Technique et Science Informatiques*, 24:1—20, 2005.

[6] F. D'Azevedo and J. Dongarra. The design and implementation of the parallel out-of-core scalapack lu, qr and cholesky factorization routines. Technical Report CS-97-347, University of Tenessee, january 1997. http://www.netlib.org.

[7] Jean-Guillaume Dumas. Efficient dot product over finite fields. In Victor G. Ganzha, Ernst W. Mayr, and Evgenii V. Vorozhtsov, editors, *Proceedings of the seventh International Workshop on Computer Algebra in Scientific Computing, Yalta, Ukraine*, pages 139–154. Technische Universität München, Germany, July 2004.

[8] Jean-Guillaume Dumas, Thierry Gautier, and Clément Pernet. Finite field linear algebra subroutines. In Teo Mora, editor, *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, Lille, France*, pages 63–74. ACM Press, New York, July 2002.

[9] Jean-Guillaume Dumas, Pascal Giorgi, and Clément Pernet. FFPACK: Finite field linear algebra package. In Jaime Gutierrez, editor, *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation, Santander, Spain*, pages 63–74. ACM Press, New York, July 2004.

[10] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2), 2005. special issue on "Program Generation, Optimization, and Adaptation".

[11] Matteo Frigo, Charles E. Leiserson, Harald Prokop, and Sridhar Ramachandran. Cache-oblivious algorithms. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 285, Washington, DC, USA, 1999. IEEE Computer Society.

[12] Alan G. Ganek and Thomas A. Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, 2003.

[13] Thierry Gautier, Gilles Villard, Jean-Louis Roch, Jean-Guillaume Dumas, and Pascal Giorgi. Givaro, une bibliothque c++ pour le calcul formel : arithmétique exacte et structures de données. Software, ciel-00000022, October 2005. `www-lmc.imag.fr/Logiciels/givaro`.

[14] Xiaohan Huang and Victor Y. Pan. Fast rectangular matrix multiplications and improving parallel matrix computations. In ACM, editor, *PASCO '97. Proceedings of the second international symposium on parallel symbolic computation, July 20–22, 1997, Maui, HI*, pages 11–23, New York, NY 10036, USA, 1997. ACM Press.

[15] Samir Jafar, Thierry Gautier, Axel W. Krings, and Jean-Louis Roch. A checkpoint/recovery model for heterogeneous dataflow computations using work-stealing. In LNCS Springer-Verlag, editor, *EUROPAR'2005*, Lisboa, Portogal, August 2005.

[16] J. Jájá. *An Introduction to Parallel Algorithms*. Addison-Wesley, Reading, Massachussets, 1992.

[17] K. H. Randall M. Frigo, C. E. Leiserson. The implementation of the cilk-5 multithreaded language. In *Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation*, pages 212–223. ACM Press, 1998.

[18] Frederic Ogel, Bertil Folliot, and Ian Piumarta. On reflexive and dynamically adaptable environments for distributed computing. In *ICDCS Workshops*, pages 112–117. IEEE Computer Society, 2003.

[19] Victor Y. Pan and Franco P. Preparata. Work-preserving speed-up of parallel matrix computations. *SIAM Journal on Computing*, 24(4), 1995.

[20] Clément Pernet. Implementation of Winograd's matrix multiplication over finite fields using ATLAS level 3 BLAS. Technical report, Laboratoire Informatique et Distribution, July 2001. `www-id.imag.fr/Apache/RR/RR011122FFLAS.ps.gz`.

[21] B. Richard, P. Augerat, N. Maillard, S. Derr, S. Martin, and C. Robert. I-cluster: Reaching top500 performance using mainstream hardware. Technical Report HPL-2001-206 20010831, HP Laboratories Grenoble, August 2001.

[22] Rob V. van Nieuwpoort, Jason Maassen, Thilo Kielmann, and Henri E. Bal. Satin: Simple and efficient java-based grid programming. *Scalable Computing: Practice and Experience*, 6(3):19–32, September 2005.

[23] R. Clint Whaley, Antoine Petitet, and Jack J. Dongarra. Automated empirical optimizations of software and the ATLAS project. *Parallel Computing*, 27(1–2):3–35, January 2001. `www.elsevier.nl/gej-ng/10/35/21/47/25/23/article.pdf`.

Van Dat Cung and Christophe Rapine
GILCO Laboratory, ENSGI-INPG, H building, Office H123.
46, avenue Félix-Viallet, 38031 Grenoble, FRANCE.
{Van-Dat.Cung,Christophe.Rapine}@gilco.inpg.fr,
gilco.inpg.fr/~{cung,rapine}.

Vincent Danjean, Thierry Gautier, Guillaume Huard, Bruno Raffin, Jean-Louis Roch and Denis Trystram
Laboratoire Informatique et Distribution, ENSIMAG - antenne de Montbonnot
ZIRST 51, avenue Jean Kuntzmann, 38330 Montbonnot Saint Martin, FRANCE.
FirstName.LastName@imag.fr, www-id.imag.fr/Membres.

Jean-Guillaume Dumas
Laboratoire de Modélisation et Calcul, Université Joseph Fourier, Grenoble I
51, av. des Mathématiques, BP 53X, 38041 Grenoble, FRANCE.
Jean-Guillaume.Dumas@imag.fr,
www-lmc.imag.fr/lmc-mosaic/Jean-Guillaume.Dumas

A Appendix

A.1 Bottom-up TRSM

We need to group the last recursive `ULeft-Trsm` call and the update of B_1. The following algorithm thus just computes these last two steps ; the first step being performed by the work stealing as shown afterwards.

Algorithm `BUT`
Input: $(A_2; A_3; B)$.
Output: X_2, k_{BUT}.

Mutual Exclusion section {
  if $(k_{TDTI} \geq k_{BUT})$ Return;
  $k_B := Choice(1..(k_{BUT} - k_{TDTI}))$.
  Split remaining columns into $k_{TDTI}..(k_{BUT}-k_B)$ *and* $(k_{BUT}-k_B)..k_{BUT}$

$$\begin{bmatrix} A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \\ & A_{3,3} \end{bmatrix} \begin{bmatrix} X_{2,1} \\ X_{2,2} \end{bmatrix} = \begin{bmatrix} B_1 \\ B_{2,1} \\ B_{2,2} \end{bmatrix}$$

  $k_{BUT} := k_{BUT} - k_B$;
}
$X_{2,2}$:=`ULeft-Trsm`$(A_{3,3}, B_{2,2})$;
$B_1 := B_1 - A_{2,2}X_{2,2}$;
$B_{2,1} := B_{2,1} - A_{3,2}X_{2,2}$;
$X_{2,1}$:=`BUT`$\left(A_{2,1}; A_{3,1}; \begin{bmatrix} B_1 \\ B_{2,1} \end{bmatrix}\right)$

A.2 Top down triangular inversion of A_1

Algorithm `TDTI`
Input: $\left(A_1^{-1}; A_2; A_3\right)$.
Output: A^{-1}, k_{TDTI}.

Mutual Exclusion section {
if ($k_{TDTI} \geq k_{BUT}$) Return;
$k_I := Choice(1..(k_{BUT} - k_{TDTI}))$.
Split remaining columns of A_2 *and* A_3 *into* $k_{TDTI}..(k_{TDTI} + k_I)$ *and* $(k_{TDTI} + k_I)..k_{BUT}$

$$\begin{bmatrix} A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \\ & A_{3,3} \end{bmatrix}$$

}
Parallel {
$A_{3,1}^{-1}$:=`Inverse(`$A_{3,1}$`)`;
$T := A_1^{-1}.A_{2,1}$
}
$A'_{2,1} = -T.A_{3,1}^{-1}$
Now, let $A_1^{'-1} = \begin{bmatrix} A_1^{-1} & A'_{2,1} \\ & A_{3,1}^{-1} \end{bmatrix}$ and $A'_2 = \begin{bmatrix} A_{2,2} \\ A_{3,2} \end{bmatrix}$
Mutual Exclusion section {
$k_{TDTI} := k_{TDTI} + k_I$;
}
$A_{3,3}^{-1}$:=`TDTI(`$A_1^{'-1}$`;`A'_2`;`$A_{3,3}$`)`;

On the complexity of the D5 principle

Xavier Dahan Marc Moreno Maza Éric Schost Yuzhen Xie

March 17, 2006

Abstract

The D5 Principle was introduced in 1985 by Jean Della Dora, Claire Dicrescenzo and Dominique Duval in their celebrated note "About a new method for computing in algebraic number fields". This innovative approach automatizes reasoning based on case discussion and is also known as "Dynamic Evaluation". Applications of the D5 Principle have been made in Algebra, Computer Algebra, Geometry and Logic.

Many algorithms for solving polynomial systems symbolically need to perform standard operations, such as GCD computations, over coefficient rings that are direct products of fields rather than fields. We show in this paper how asymptotically fast algorithms for polynomials over fields can be adapted to this more general context, thanks to the D5 Principle.

1 Introduction

The standard approach for computing with an algebraic number is through the data of its irreducible minimal polynomial over some base field k. However, in typical tasks such as polynomial system solving, involving many algebraic numbers of high degree, following this approach will require using probably costly factorization algorithms. Jean Della Dora, Claire Dicrescenzo and Dominique Duval introduced "Dynamic Evaluation" techniques (also termed "D5 Principle") as a means to compute with algebraic numbers, while avoiding factorization. Roughly speaking, this approach leads one to compute over *direct products of field extensions of* k, instead of only field extensions.

Applications of Dynamic Evaluation have been made by many authors: González-López and Recio (1993), Gómez Díaz (1994), Duval (1994), Lombardi (2003) and others. Many algorithms for polynomial system solving rely on this philosophy; see, for instance, the work of Lazard (1992), Kalkbrener (1993), Dellière (1999), Moreno Maza (2000), Mora (2003). Boulier et al. (2006).

This work is aiming at filling the lack of complexity results for this approach. The addition and multiplication over a direct product of fields are easily proved to be *quasi-linear* (in a natural complexity measure). As for the inversion, it has to be replaced by *quasi-inversion*: following the D5 philosophy, meeting zero-divisors in the computation will lead to *splitting* the direct product of fields into a family thereof. It is much more tricky to prove quasi-linear complexity estimate for quasi-inversion, because the algorithm relies on

other algorithms, for which such an estimate has to be proved: the GCD and the splitting algorithms.

In this article, direct product of fields will be described using *triangular sets*. In what follows, we assume that the base field k is perfect.

Definition 1.1. A *triangular set* T is a family of n-variate polynomials over k:

$$T \ = (T_1(X_1)\ ,\ \ T_2(X_1, X_2)\ ,\ \ldots\ ,\ \ T_n(X_1, \ldots, X_n))\,,$$

which forms a reduced Gröbner basis for the lexicographic order induced by $X_n > \cdots > X_1$, and such that the ideal $\langle T\rangle$ generated by T in $k[X_1, \ldots, X_n]$ is radical.

If T is a triangular set, the residue class ring $\mathbb{K}(T) := k[X_1, \ldots, X_n]/\langle T\rangle$ is a direct product of fields. Hence, our questions can be basically rephrased as studying the complexity of operations (addition, multiplication, quasi-inversion) modulo triangular sets. The following notation helps us quantify the complexity of these algorithms.

Definition 1.2. We denote by $\deg_i(T)$ the degree of T_i in X_i, for all $1 \leq i \leq n$, and by $\deg(T)$ the product $\deg_1(T) \cdots \deg_n(T)$. We call it the *degree* of T.

Observe that $\langle T\rangle$ is zero-dimensional and that for all $1 \leq i \leq n$, the set $(T_1 \ldots, T_i)$ is a triangular set of $k[X_1, \ldots, X_i]$. The zero-set of T in the affine space $\mathbb{A}^n(\bar{k})$ has a particular feature: it is *equiprojectable* (Aubry and Valibouze, 2000; Dahan and Schost, 2004); besides, its cardinality equals $\deg(T)$.

Definition 1.3. A *triangular decomposition* of a zero-dimensional radical ideal $I \subset k[X_1, \ldots, X_n]$ is a family $\mathbf{T} = T^1, \ldots, T^e$ of triangular sets, such that $I = \langle T^1\rangle \cap \cdots \cap \langle T^e\rangle$ and $\langle T^i\rangle + \langle T^j\rangle = \langle 1\rangle$ for all $i \neq j$. A triangular decomposition $\mathbf{T}'$ of I *refines* another decomposition $\mathbf{T}$ if for every $T \in \mathbf{T}$ there exists a (necessarily unique) subset $\mathrm{decomp}(T, \mathbf{T}') \subseteq \mathbf{T}'$ which is a triangular decomposition of $\langle T\rangle$.

Let T be a triangular set, let $\mathbf{T} = T^1, \ldots, T^e$ be a triangular decomposition of $\langle T\rangle$, and define $\mathbb{K}(\mathbf{T}) := \mathbb{K}(T^1) \times \cdots \times \mathbb{K}(T^e)$. Then by the Chinese remainder theorem, $\mathbb{K}(T) \simeq \mathbb{K}(\mathbf{T})$. Now let $\mathbf{T}'$ be a refinement of $\mathbf{T}$. For each triangular set T^i in $\mathbf{T}$, denote by $U^{i,1}, \ldots, U^{i,e_i}$ the triangular sets in $\mathrm{decomp}(T^i, \mathbf{T}')$. We have the following e isomorphism:

$$\phi_i :\ \mathbb{K}(T^i) \simeq \mathbb{K}(U^{i,1}) \times \cdots \times \mathbb{K}(U^{i,e_i}), \tag{1}$$

which extend to the following e isomorphisms, where y is a new variable.

$$\Phi_i :\ \mathbb{K}(T^i)[y] \simeq \mathbb{K}(U^{i,1})[y] \times \cdots \times \mathbb{K}(U^{i,e_i})[y]. \tag{2}$$

Definition 1.4. For $\mathbf{h} = (h_1, \ldots, h_e) \in \mathbb{K}(T^1)[y] \times \cdots \times \mathbb{K}(T^e)[y]$, we call *split* of $\mathbf{h}$ with respect to $\mathbf{T}$ and $\mathbf{T}'$, and write $\mathrm{split}(\mathbf{h}, \mathbf{T}, \mathbf{T}')$ the vector $(\Phi_1(h_1), \ldots, \Phi_e(h_e))$.

Note that if $g \in \mathbb{K}(T)[y]$, then we have $\mathrm{split}(g, \{T\}, \mathbf{T}') = \mathrm{split}(\mathrm{split}(g, \{T\}, \mathbf{T}), \mathbf{T}')$. For simplicity, we define $\mathrm{split}(g, \mathbf{T}) = \mathrm{split}(g, \{T\}, \mathbf{T})$.

We now introduce a fundamental notion, that of *non-critical* decompositions. It is motivated by the following remark. Let $\mathbf{T} = T^1, \ldots, T^e$ be a family of triangular sets, with $T^j = (T_1^j, T_2^j, \ldots, T_n^j)$. For $1 \leq i \leq n$, we write $T_{\leq i}^j = T_1^j\ ,\ T_2^j\ ,\ \ldots\ ,\ T_i^j$ and define the family $\mathbf{T}_{\leq i}$ by:

$$\mathbf{T}_{\leq i} = \{T_{\leq i}^j \ \mid \ j \leq e\} \quad \text{(with no repetition allowed)}.$$

Even if $\mathbf{T}$ is a triangular decomposition of a 0-dimensional radical ideal $I \subset k[X_1, \ldots, X_n]$, $\mathbf{T}_{\leq i}$ is not necessarily a triangular decomposition of $I \cap k[X_1, \ldots, X_i]$. Indeed, with $n = 2$ and $e = 2$, consider $T^1 = ((X_1 - 1)(X_1 - 2), X_2)$ and $T^2 = ((X_1 - 1)(X_1 - 3), X_2 - 1)$. The family $\mathbf{T} = T^1, T^2$ is a triangular decomposition of the ideal $I = \langle T^1 \rangle \cap \langle T^2 \rangle$. However, the family of triangular sets

$$\mathbf{T}_{\leq 1} = \{T_1^1 = (X_1 - 1)(X_1 - 2),\ T_2^1 = (X_1 - 1)(X_1 - 3)\}$$

is not a triangular decomposition of $I \cap k[X_1]$ since $\langle T_1^1 \rangle + \langle T_1^2 \rangle = \langle X_1 - 1 \rangle$.

Definition 1.5. Let T be a triangular set in $k[X_1, \ldots, X_n]$. Two polynomials $a, b \in \mathbb{K}(T)[y]$ are *coprime* if the ideal $\langle a, b \rangle \subset \mathbb{K}(T)[y]$ equals $\langle 1 \rangle$.

Definition 1.6. Let $T \neq T'$ be two triangular sets, with $T = (T_1, \ldots, T_n)$ and $T' = (T'_1, \ldots, T'_n)$. The least integer ℓ such that $T_\ell \neq T'_\ell$ is called the *level* of the pair $\{T, T'\}$. The pair $\{T, T'\}$ is *critical* if T_ℓ and T'_ℓ are not coprime in $k[X_1, \ldots, X_{\ell-1}]/\langle T_1, \ldots, T_{\ell-1} \rangle[X_\ell]$. A family of triangular sets $\mathbf{T}$ is *non-critical* if it has no critical pairs, otherwise it is said to be *critical*.

The pair $\{T^1, T^2\}$ in the above example has level 1 and is critical. Consider $U^{1,1} = (X_1 - 1, X_2)$, $U^{1,2} = (X_1 - 2, X_2)$, $U^{2,1} = (X_1 - 1, X_2 - 1)$ and $U^{2,2} = (X_1 - 3, X_2 - 1)$. Observe that $\mathbf{U} = \{U^{1,1}, U^{1,2}, U^{2,1}, U^{2,2}\}$ is a non-critical triangular decomposition of I refining $\{T^1, T^2\}$ and that $\mathbf{U}_{\leq 1}$ is a triangular decomposition $I \cap k[X_1, X_2]$.

This notion of critical pair is fundamental. In fact, fast algorithms for the innocuous splitting operations Φ_i of Equation (2) are not guaranteed for critical decompositions, as shown in the following extension of the previous example. Consider a third triangular set $T^3 = ((X_1 - 2)(X_1 - 3), X_2 + X_1 - 3)$. One checks that $\mathbf{V} = \{T^1, T^2, T^3\}$ is a triangular decomposition of $T = ((X_1 - 1)(X_1 - 2)(X_1 - 3), X_2(X_2 - 1))$. However, splitting an element p from $\{T\}$ to $\mathbf{V}$ requires to compute

$$p \bmod (X_1 - 1)(X_1 - 2),\ p \bmod (X_1 - 1)(X_1 - 3),\ p \bmod (X_1 - 2)(X_1 - 3),$$

whence some redundancies. In general, these redundancies prevent the splitting computation from being quasi-linear w.r.t. $\deg(T)$. But if the triangular decomposition is non-critical, then there is no more redundancy, and the complexity of splitting p can be hoped to be quasi-linear.

Removing critical pairs of a critical triangular decomposition in order to be able to split fast requires to delete the common factors between the polynomials involved in the

decomposition. To do it fast (in quasi-linear time), the *coprime factorization* or *gcd-free basis computation* algorithm is used. Of course to implement this algorithm over a direct product of fields, one first need to be able to compute GCD's over such a product in quasi-linear time.

Since $\mathbb{K}(T)$ is a direct product of fields, any pair of univariate polynomials $f, g \in \mathbb{K}(T)[y]$ admits a GCD h in $\mathbb{K}(T)[y]$, in the sense that the ideals $\langle f, g\rangle$ and $\langle h\rangle$ coincide, see Moreno Maza and Rioboo (1995). However, even if f, g are both monic, there may not exist a monic polynomial h in $\mathbb{K}(T)[y]$ such that $\langle f, g\rangle = \langle h\rangle$ holds: consider for instance $f = y + \frac{a+1}{2}$ (assuming that 2 is invertible in k) and $g = y + 1$ where $a \in \mathbb{K}(T)$ satisfies $a^2 = a$, $a \neq 0$ and $a \neq 1$. GCD's with non-invertible leading coefficients are of limited practical interest; this leads us to the following definition.

Definition 1.7. Let f, g be in $\mathbb{K}(T)[y]$. An *extended greatest common divisor (XGCD)* of f and g is a sequence $((h_i, u_i, v_i, T^i), 1 \leq i \leq e)$, where $\mathbf{T} = T^1, \ldots, T^e$ is a non-critical decomposition of T and for all $1 \leq i \leq e$, h_i, u_i, v_i are polynomials in $\mathbb{K}(T^i)[y]$, such that the following holds. Let $f_1, \ldots, f_e = \text{split}(f, \{T\}, \mathbf{T})$ and $g_1, \ldots, g_e = \text{split}(g, \{T\}, \mathbf{T})$; then for $1 \leq i \leq e$, we have:

- h_i is monic or null,
- the inequalities $\deg u_i < \deg g_i$ and $\deg v_i < \deg f_i$ hold,
- the equalities $\langle f_i, g_i\rangle = \langle h_i\rangle$ and $h_i = u_i f_i + v_i g_i$ hold.

One easily checks that such XGCD's exists, and can be computed, for instance by applying the D5 Principle to the Euclidean algorithm. To compute GCD's in quasi-linear time over a direct product of fields, we will actually adapt the *Half-GCD* techniques (Yap, 1993) in Section 4.

Our last basic ingredient is the suitable generalization of the notion of inverse to direct products of fields.

Definition 1.8. A *quasi-inverse* of an element $f \in \mathbb{K}(T)$ is a sequence of couples $((u_i, T^i), 1 \leq i \leq e)$ where $\mathbf{T} = T^1, \ldots, T^e$ is a non-critical decomposition of T and u_i is an element of $\mathbb{K}(T^i)$ for all $1 \leq i \leq e$, such that the following holds. Let $f_1, \ldots, f_e = \text{split}(f, \{T\}, \mathbf{T})$; then for $1 \leq i \leq e$ we have either $f_i = u_i = 0$, or $f_i u_i = 1$.

Obtaining fast algorithms for GCD's, quasi-inverses and removal of critical pairs requires a careful inductive process that we summarize in this paper.

- We first need complexity estimates for multiplication modulo a triangular set and splitting w.r.t. triangular decompositions. This is done in Section 3.
- Assuming that multiplications and quasi-inverse computations can be computed fast in $\mathbb{K}(T)$, and assuming that we can remove critical pairs from critical triangular decompositions of $\langle T\rangle$, we obtain in Section 4 a fast algorithm for computing GCD's in $\mathbb{K}(T)[y]$. Note that Langemyr (1991) states that GCD's over products of fields can be computed in quasi-linear time, but with no proof.

- Assuming that GCD's can be computed fast in $\mathbb{K}(T_1, \ldots, T_{n-1})[X_n]$, we present fast algorithms for quasi-inverses in $\mathbb{K}(T)$ (Section 5), coprime factorization for polynomials in $\mathbb{K}(T_1, \ldots, T_{n-1})[X_n]$ (Section 6) and refining a triangular decomposition $\mathbf{T}$ of T into a non-critical one (Section 7).

These are the basic blocks for our inductive process, which yields our main results:

Theorem 1.9. *For any $\varepsilon > 0$, there exists $A_\varepsilon > 0$ such that addition, multiplication and quasi-inversion in $\mathbb{K}(T)$ can be computed in $A_\varepsilon^n \deg(T)^{1+\varepsilon}$ operations in k.*

Theorem 1.10. *There exists $G > 0$, and for any $\varepsilon > 0$, there exists $A_\varepsilon > 0$, such that one can compute an extended greatest common divisor of polynomials in $\mathbb{K}(T)[y]$, with degree at most d, using at most $G\, A_\varepsilon^n\, d^{1+\varepsilon} \deg(T)^{1+\varepsilon}$ operations in k.*

Due to space constraints, it is not possible to give all details of our algorithms in this paper. Hence, some algorithms like GCD receive a detailed treatment, while we have to be more sketchy on other ones.

2 Complexity notions

We start by recalling basic results for operations on univariate polynomials.

Definition 2.1. A *multiplication time* is a map $\mathsf{M} : \mathbb{N} \to \mathbb{R}$ such that:

- For any ring R, polynomials of degree less than d in $R[X]$ can be multiplied in at most $\mathsf{M}(d)$ operations $(+, \times)$ in R.
- For any $d \leq d'$, the inequalities $\frac{\mathsf{M}(d)}{d} \leq \frac{\mathsf{M}(d')}{d'}$ and $\mathsf{M}(dd') \leq \mathsf{M}(d)\mathsf{M}(d')$ hold.

Note that in particular that the inequalities $\mathsf{M}(d) \geq d$ and $\mathsf{M}(d) + \mathsf{M}(d') \leq \mathsf{M}(d + d')$ hold for all d, d'. Using the result of Cantor and Kaltofen (1991), that follows the work of Schönhage and Strassen, we know that there exists $c \in \mathbb{R}$ such that the function $d \mapsto c\, d \log\mathrm{p}(d) \log\mathrm{p} \log\mathrm{p}(d)$ is a multiplication time. In what follows, the function logp is defined by $\log\mathrm{p}(x) = 2 \log_2(\max\{2, x\})$: this function turns out to be more convenient than the classical logarithm for handling inequalities.

Fast polynomial multiplication is the basis of many other fast algorithms: Euclidean division, computation of the subproduct tree (see Chapter 10 in von zur Gathen and Gerhard (1999) and Section 6 of this article), and multiple remaindering.

Proposition 2.2. *There exists a constant $C \geq 1$ such that the following holds over any ring R. Let M be a multiplication time. Then:*

1. *Dividing in $R[X]$ a polynomial of degree less than $2d$ by a monic polynomial of degree at most d requires at most $5\mathsf{M}(d) + O(d) \leq C\, \mathsf{M}(d)$ operations $(+, \times)$ in R.*
2. *Let F be a monic polynomial of degree d in $R[X]$. Then additions and multiplications in $R[X]/F$ requires at most $6\, \mathsf{M}(d) + O(d) \leq C\, \mathsf{M}(d)$ operations $(+, \times)$ in R.*

3. *Let $F_1, \ldots, F_s$ be non-constant monic polynomials in $R[X]$, with sum of degrees d. Then one can compute the subproduct tree associated to $F_1, \ldots, F_s$ using at most $\mathsf{M}(d)\log(d)$ operations $(+, \times)$ in R.*

4. *Let $F_1, \ldots, F_s$ be non-constant monic polynomials in $R[X]$, with sum of degrees d. Then given A in $R[X]$ of degree less than d, one can compute $A \bmod F_1, \ldots, A \bmod F_s$ within $11\,\mathsf{M}(d)\log(d) + O(d\log(d)) \leq C\,\mathsf{M}(d)\log(d)$ operations $(+, \times)$ in R.*

5. *Assume that R is a field. Then, given two polynomials in $R[X]$ of degree at most d, computing their monic GCD and their Bézout coefficients can be done in no more than $33\,\mathsf{M}(d)\log(d) + O(d\log(d)) \leq C\,\mathsf{M}(d)\log(d)$ operations $(+, \times, /)$ in R.*

6. *Assume that R is a field and that F is a monic squarefree polynomial in $R[X]$ of degree d. Then, computing a quasi-inverse modulo F of a polynomial $G \in R[X]$ of degree less than d can be done in no more than $71\,\mathsf{M}(d)\log(d) + O(d\log(d)) \leq C\,\mathsf{M}(d)\log(d)$ operations $(+, \times, /)$ in R.*

PROOF. The first point is proved in Theorem 9.6 of (von zur Gathen and Gerhard, 1999) and implies the second one. The third and fourth points are proved in Lemma 10.4 and Theorem 10.15 of the same book. The fifth point is reported in Theorem 11.5 of that book (with a better constant), and is a particular case of Section 4 of this article. If F has no multiple factors in $R[X]$, a quasi-inverse of G modulo F can be obtained by at most two extended GCD computations and one division with entries of degree at most d. Using estimates for the GCD leads to the result claimed in point 6. □

We now define our key complexity notion, arithmetic time for triangular sets.

Definition 2.3. An *arithmetic time* is a function $T \mapsto \mathsf{A}_n(T)$ with real positive values and defined over all triangular sets in $k[X_1, \ldots, X_n]$ such that the following conditions hold.

(E_0) For every triangular decomposition $\mathbf{T} = T^1, \ldots, T^e$ of T, we have $\mathsf{A}_n(T^1) + \cdots + \mathsf{A}_n(T^e) \leq \mathsf{A}_n(T)$.

(E_1) Every addition or multiplication in $\mathbb{K}(T)$ can be done in at most $\mathsf{A}_n(T)$ operations in k.

(E_2) Every quasi-inverse in $\mathbb{K}(T)$ can be computed in at most $\mathsf{A}_n(T)$ operations in k.

(E_3) Given a triangular decomposition $\mathbf{T}$ of T, one can compute a *non-critical* triangular decomposition $\mathbf{T}'$ which refines $\mathbf{T}$, in at most $\mathsf{A}_n(T)$ operations in k.

(E_4) For every $\alpha \in \mathbb{K}(T)$ and every non-critical triangular decomposition $\mathbf{T}$ of T, one can compute $\mathrm{split}(\alpha, \{T\}, \mathbf{T})$ in at most $\mathsf{A}_n(T)$ operations in k.

Our main goal in this paper is then to give estimates for arithmetic times. This is done through an inductive proof; the following proposition gives such a result for the base case, triangular sets in one variable.

Proposition 2.4. *If $n = 1$, then $T \in k[X_1] \mapsto C\,\mathsf{M}(\deg T)\log\mathrm{p}(\deg T)$ is an arithmetic time.*

Proof. A triangular set in one variable is simply a squarefree monic polynomial in $k[X_1]$. Hence, (E_1), (E_2) and (E_4) respectively follow from points 2, 6 and 4 in Proposition 2.2. Property (E_0) is clear. Since $n = 1$, all triangular decompositions are non-critical, and (E_3) follows. □

3 Basic complexity results: multiplication and splitting

This section is devoted to give first complexity results for triangular sets: we give upper bounds on the cost of multiplication, and splitting. In general, we do not know how to perform this last operation in quasi-linear time; however, when the decomposition is non-critical, quasi-linearity can be reached.

Proposition 3.1. *Let M be a multiplication function, and let C be the constant from Proposition 2.2. Let T be a triangular set in $k[X_1, \ldots, X_n]$. Then:*

- *Additions and multiplications modulo T can be done in at most $C^n \prod_{i\le n} \mathsf{M}(\deg_i T)$ operations in k.*
- *If $\mathbf{T}$ is a non-critical decomposition of T, then for any h in $\mathbb{K}(T)$, $\mathrm{split}(h, \{T\}, \mathbf{T})$ can be computed in at most $n\,C^n \prod_{i\le n} \mathsf{M}(\deg_i T)\log\mathrm{p}(\deg_i T)$ operations in k.*

Proof. The first part of the proposition is easy to deal with: the case of additions is obvious, using the inequality $\mathsf{M}(d) \ge d$; as to multiplication, an easy induction using point (1) in Proposition 2.2 gives the result. The end of the proof uses point (4) in Proposition 2.2; the non-critical assumption is then used through the following lemma. □

Lemma 3.2. *Consider a non-critical decomposition $\mathbf{T}$ of the triangular set $T = (T_1, \ldots, T_n)$. Write $\mathbf{T}_{\le n-1} = \{U^1, \ldots, U^s\}$, and, for all $i \le s$, denote by $T^{i,1}, \ldots, T^{i,e_i}$ the triangular sets in $\mathbf{T}$ such that $T^{i,j} \cap k[X_1, \ldots, X_{n-1}] = U^i$ (thus $\mathbf{T}$ is the set of all $T^{i,j}$, with $i \le s$ and $j \le e_i$). Then $\mathbf{T}_{\le n-1}$ is a non-critical decomposition of the triangular set $(T_1, \ldots, T_{n-1})$. Moreover, for all $i \le s$, we have:*

$$\sum_{j\le e_i} \deg_n T^{i,j} = \deg_n T.$$

As an illustration, consider again, for $n = 2$, the triangular sets

$$\begin{aligned} T^1 &= ((X_1-1)(X_1-2),\ X_2) \\ T^2 &= ((X_1-1)(X_1-3),\ X_2-1) \\ \text{and } T^3 &= ((X_1-2)(X_1-3),\ X_2+X_1-3). \end{aligned}$$

These triangular sets form a critical decomposition $\mathbf{T}$ of the ideal $\langle T^1\rangle \cap \langle T^2\rangle \cap \langle T^3\rangle$, which is also generated by $T = ((X_1-1)(X_1-2)(X_1-3), X_2(X_2-1))$.

Here, $\mathbf{T}_{\leq 1}$ is given by $\{U^1, U^2, U^3\} = \{(X_1-1)(X_1-2), (X_1-1)(X_1-3), (X_1-1)(X_1-3)\}$, so that $s = 3$. Take for instance $U^1 = (X_1-1)(X_1-2)$; then we have $e_1 = 1$ and $T^{1,e_1} = T^1$. Note then that $\deg_2 T^{1,e_1} = 1$ differs from $\deg_2 T = 2$, so the conclusion of the previous lemma is indeed violated.

4 Fast GCD computations modulo a triangular set

GCD's of univariate polynomials over a field can be computed in quasi-linear time by means of the *Half-GCD* algorithm (Brent et al., 1980; Yap, 1993). We show how to adapt this technique over the direct product of fields $\mathbb{K}(T)$ and how to preserve its complexity class. Throughout this section, we consider an arithmetic time $T \mapsto \mathsf{A}_n(T)$ for triangular sets in $k[X_1, \ldots, X_n]$.

Proposition 4.1. *For all $a, b \in \mathbb{K}(T)[y]$ with $\deg a$, $\deg b \leq d$, one can compute an extended greatest common divisor of a and b in $O(\mathsf{M}(d)\log(d))\mathsf{A}_n(T)$ operations in k.*

We prove this result by describing our GCD algorithm over the direct product of fields $\mathbb{K}(T)$ and its complexity estimate. We start with two auxiliary algorithms.

Monic forms. Any polynomial over a field can be made monic by division through its leading coefficient. Over a product of fields, this division may induce splittings. We now study this issue.

Definition 4.2. A *monic form* of $f \in \mathbb{K}(T)[y]$ is a sequence of quadruples $((u_i, v_i, m_i, T_i), 1 \leq i \leq e)$, where $\mathbf{T} = T^1, \ldots, T^e$ is a non-critical decomposition of T, u_i, v_i are in $\mathbb{K}(T^i)$ and m_i is in $\mathbb{K}(T^i)[y]$ for all $1 \leq i \leq e$, and such that the following holds.

Let $f_1, \ldots, f_e = \text{split}(f, \{T\}, \mathbf{T})$. Denote by $\text{lc}(f_i)$ the leading coefficient of f_i. Then, for all $1 \leq i \leq e$ we have $u_i = \text{lc}(f_i)$, and $m_i = v_i f_i$, and either $u_i = v_i = 0$ or $u_i v_i = 1$.

Observe that for all $1 \leq i \leq e$, the polynomial m_i is monic or null.

The following algorithm shows how to compute a monic form. This function uses a procedure quasiInverse($\mathbf{f}, \mathbf{T}$). This procedure takes as input a triangular decomposition $\mathbf{T} = T^1, \ldots, T^e$ of T and a sequence $\mathbf{f} = f_1, \ldots, f_e$ in $\mathbb{K}(T^1)[y] \times \cdots \times \mathbb{K}(T^e)[y]$ and returns a sequence $(((f_{ij}, T^{ij}), 1 \leq j \leq e_i), 1 \leq i \leq e)$ where $((f_{ij}, T^{ij}), 1 \leq j \leq e_i)$ is a quasi-inverse of f_i modulo T^i and such that $(T^{ij}, 1 \leq j \leq e_i, 1 \leq i \leq e)$ is a non-critical refinement of $\mathbf{T}$. Its complexity is studied in Section 5.

The number at the end of a line, multiplied by $\mathsf{A}_n(T)$, gives an upper bound for the total time spent at this line. Therefore, the following algorithm computes a monic form of f in at most $(8d+6)\mathsf{A}_n(T)$ operations in k.

```
monic(f, T) ==
  1   T := {T}
  2   v := (0)
  3   g := f
```

4 **while** $g \neq 0$ **repeat**
4.1 $\mathbf{u} := \text{split}(\text{lc}(g), \{T\}, \mathbf{T})$ $[d+1]$
4.2 $(\mathbf{w}, \mathbf{T}') := \text{quasiInverse}(\mathbf{u}, \mathbf{T})$ $[3d+3]$
4.3 $\mathbf{v} := \text{split}(\mathbf{v}, \mathbf{T}, \mathbf{T}')$ $[d+1]$
4.4 **for** $1 \leq i \leq \#\mathbf{v}$ **repeat**
4.4.1 **if** $v_i = 0$ **then** $v_i := w_i$ $[d+1]$
4.5 $\mathbf{T} := \mathbf{T}'$
4.6 $g := g - \text{leadingTerm}(g)$
5 $\mathbf{f} := \text{split}(f, \{T\}, \mathbf{T})$ $[d]$
6 $\mathbf{u} := \text{lc}(\mathbf{f})$
7 $\mathbf{m} := \mathbf{v} \cdot \mathbf{f}$ $[d]$
8 **return** $((u_i, v_i, m_i, T^i), 1 \leq i \leq \#\mathbf{T})$

Division with monic remainder. The previous notion can then be used to compute Euclidean divisions, producing *monic* remainders: they will be required in our fast Euclidean algorithm for XGCD's.

Definition 4.3. Let $f, g \in \mathbb{K}(T)[y]$ with g monic. A *division with monic remainder* of f by g is a sequence of tuples $((g_i, q_i, v_i, u_i, r_i, T^i), 1 \leq i \leq e)$ such that $\mathbf{T} = T^1, \ldots, T^e$ is a non-critical decomposition of T, and, for all $1 \leq i \leq e$, we have $u_i, v_i \in \mathbb{K}(T^i)$ and $g_i, q_i, r_i, \in \mathbb{K}(T^i)[y]$, and such that the following holds.

Let $f_1, \ldots, f_e = \text{split}(f, \{T\}, \mathbf{T})$ and $g_1, \ldots, g_e = \text{split}(g, \{T\}, \mathbf{T})$. Then, for all $1 \leq i \leq e$, the polynomial r_i is null or monic, we have either $u_i = v_i = 0$ or $u_i v_i = 1$, and the polynomials q_i and $u_i r_i$ are the quotient and remainder of f_i by g_i in $\mathbb{K}(T^i)[y]$.

The following algorithm computes a division with monic remainder of f by g and requires at most $(5\mathsf{M}(d) + O(d))\mathsf{A}_n(T)$ operations in k. We write $(q, r) = \text{div}(f, g)$ for the quotient and the remainder in the (standard) division with remainder in $\mathbb{K}(T)[y]$.

$\text{mdiv}(f, g, T)$ ==
1 $(q, r) := \text{div}(f, g)$ $[5\mathsf{M}(d) + O(d)]$
2 $((u_i, v_i, r_i, T^i), 1 \leq i \leq \#\mathbf{T}) := \text{monic}(r, T)$ $[8d-2]$
3 $(q_i, 1 \leq i \leq \#\mathbf{T}) := \text{split}(q, \{T\}, \mathbf{T})$ $[d+1]$
4 $(g_i, 1 \leq i \leq \#\mathbf{T}) := \text{split}(g, \{T\}, \mathbf{T})$ $[d]$
5 **return** $((g_i, q_i, u_i, v_i, T^i), 1 \leq i \leq \#\mathbf{T})$

XGCD's. We are now ready to generalize the *Half-Gcd* method as exposed in Yap (1993). We introduce the following operations. For $a, b \in \mathbb{K}(T)[y]$ with $0 < \deg b < \deg a = d$, each of the following algorithms $\text{M}_{\text{gcd}}(a, b, T)$ and $\text{M}_{\text{hgcd}}(a, b, T)$ returns a sequence $((M_1, T^1), \ldots, (M_e, T^e))$ where

(s_1) $\mathbf{T} = T^1, \ldots, T^e$ is a non-critical triangular decomposition of T,

(s_2) M_i is a square matrix of order 2 with coefficients in $\mathbb{K}(T^i)[y]$,

such that, if we define $(a_1, \ldots, a_e) = \text{split}(a, \{T\}, \mathbf{T})$ and $(b_1, \ldots, b_e) = \text{split}(b, \{T\}, \mathbf{T})$, then, for all $1 \leq i \leq e$, defining $(t_i, s_i) = (a_i, b_i)\ {}^tM_i$, we have

(s_3) in the case of $\mathrm{M}_{\mathrm{gcd}}$, the polynomial t_i is a GCD of a_i, b_i and $s_i = 0$ holds,

(s'_3) in the case of $\mathrm{M}_{\mathrm{hgcd}}$, the ideals $\langle t_i, s_i\rangle$ and $\langle a_i, b_i\rangle$ of $\mathbb{K}(T^i)[y]$ are identical, and $\deg s_i < \lceil d/2\rceil \leq \deg t_i$ holds.

The algorithm below implements $\mathrm{M}_{\mathrm{gcd}}(a, b, T)$, and is an extension of the analogue algorithm known over fields. Observe that if the input triangular set T is not decomposed during the algorithm, in particular if $\mathbb{K}(T)$ is a field, then the algorithm yields generators of the ideal $\langle a, b\rangle$. If T is decomposed, then the lines from 5 to 7.3.1 guarantee that $\mathrm{M}_{\mathrm{gcd}}(a, b, T)$ generates a non-critical triangular decomposition of T.

$\mathrm{M}_{\mathrm{gcd}}(a, b, T)$ ==

0 $\mathbf{G} := [\,]$; $\mathbf{T} := [\,]$;

1 $((M_i, T^i), 1 \leq i \leq e) := \mathrm{M}_{\mathrm{hgcd}}(a, b, T)$ $[H(d)]$

2 $(a_1, \ldots, a_e) := \text{split}(a, (T^i, 1 \leq i \leq e))$ $[O(d)]$

3 $(b_1, \ldots, b_e) := \text{split}(b, (T^i, 1 \leq i \leq e))$ $[O(d)]$

4 **for** i **in** $1 \cdots e$ **repeat**

4.1 $(t_i, s_i) := (a_i, b_i)\ {}^tM_i$ $[4\,\mathsf{M}(d) + O(d)]$

4.2 **if** $s_i = 0$ **then**

4.2.1 $\mathbf{G} := \mathbf{G}, (M_i, T^i)$

4.2.2 $\mathbf{T} := \mathbf{T}, T^i$

4.3 $((s_{ij}, q_{ij}, r_{ij}, u_{ij}, v_{ij}, T^{ij}), 1 \leq j \leq e_i) := \text{mdiv}(t_i, s_i)$ $[\frac{5}{2}\mathsf{M}(d) + O(d)]$

4.4 $(M_{ij}, 1 \leq j \leq e_i) := \text{split}(M_i, (T^{ij}, 1 \leq j \leq e_i))$ $[O(d)]$

4.5 **for** j **in** $1 \cdots e_i$ **repeat**

4.5.1 $M_{ij} := \begin{pmatrix} 0 & 1 \\ v_{ij} & -q_{ij}v_{ij} \end{pmatrix} M_{ij}$ $[2\,\mathsf{M}(d) + O(d)]$

4.5.2 **if** $r_{ij} = 0$ **then**

4.5.2.1 $\mathbf{G} := \mathbf{G}, (M_{ij}, T^i)$

4.5.2.2 $\mathbf{T} := \mathbf{T}, T^{ij}$

4.5.3 $((N_{ijk}, T^{ijk}), 1 \leq k \leq e_{ij}) := \mathrm{M}_{\mathrm{gcd}}(s_{ij}, r_{ij}, T^{ij})$ $[G(d/2)]$

4.5.4 $(M_{ijk}, 1 \leq k \leq e_{ij}) := \text{split}(M_{ij}, (T^{ijk}, 1 \leq k \leq e_{ij}))$ $[O(d)]$

4.5.5 **for** k **in** $1 \cdots e_{ij}$ **repeat**

4.5.5.1 $M_{ijk} := N_{ijk}M_{ijk}$ $[8\,\mathsf{M}(d) + O(d)]$

4.5.5.2 $\mathbf{G} := \mathbf{G}, (M_{ijk}, T^{ijk})$

4.5.5.3 $\mathbf{T} := \mathbf{T}, T^{ijk}$

5 $\mathbf{T}' := \text{removeCriticalPairs}(\mathbf{T})$ $[1]$

6 **Res** := [ ]

7 **for** $(M, T) \in \mathbf{G}$ **repeat**

7.1 $\mathsf{U} := \text{decomp}(T, \mathbf{T}')$

7.2 $(M_\ell, 1 \leq \ell \leq \#\mathsf{U}) := \text{split}(M, \{T\}, \mathsf{U})$ $[O(d)]$

7.3 **for** $1 \leq \ell \leq \#\mathsf{U}$ **do**
7.3.1 $\quad$ **Res** := **Res**, (M_i, U^i)
8 **return Res**

The Half-GCD algorithm can be adapted to $\mathbb{K}(T)[y]$ (not reported here due to space consideration) leading to an implementation of $\mathrm{M}_{\mathrm{hgcd}}(a, b, T)$. It has a structure very similar to $\mathrm{M}_{\mathrm{gcd}}(a, b, T)$, see (Yap, 1993) for details in the case when the coefficients lie in a field.

Now, we give running time estimates for $\mathrm{M}_{\mathrm{hgcd}}(a, b, T)$ and $\mathrm{M}_{\mathrm{gcd}}(a, b, T)$. For $0 < \deg b < \deg a = d$, we denote by $G(d)$ and $H(d)$ respective upper bounds for the running time of $\mathrm{M}_{\mathrm{gcd}}(a, b)$ and $\mathrm{M}_{\mathrm{hgcd}}(a, b)$, in the sense that both operations can be done in respective times $G(d)\mathsf{A}_n(T)$ and $H(d)\mathsf{A}_n(T)$.

The number at the end of an above line, multiplied by $\mathsf{A}_n(T)$, gives an upper bound of the running time of this line. These estimates follow from the super-linearity of the arithmetic time for triangular sets, the running time estimates of the operation $\mathrm{mdiv}(f, g, T)$ and classical degree bounds for the intermediate polynomials in the Extended Euclidean Algorithms; see for instance Chapter 3 in (von zur Gathen and Gerhard, 1999). Therefore, counting precisely the degrees appearing, we have: $G(d) \leq G(d/2) + H(d) + (33/2)\mathsf{M}(d) + O(d)$. The operation $\mathrm{M}_{\mathrm{hgcd}}(a, b, T)$ makes two recursive calls with input polynomials of degree at most $d/2$, leading to $H(d) \leq 2H(d/2) + (33/2)\mathsf{M}(d) + O(d)$. The superlinearity of M implies

$$H(d) \leq \frac{33}{2}\mathsf{M}(d)\ \log d + O(d\ \log\ d) \quad \text{and} \quad G(d) \leq 2H(d) + 2\mathsf{M}(d) + O(d).$$

This leads to the result reported in Proposition 4.1.

We conclude with a specification of a function used in the remaining sections. For a triangular decomposition $\mathbf{T} = T^1, \ldots, T^e$ of T, two sequences $\mathbf{f} = f_1, \ldots, f_e$ and $\mathbf{g} = g_1, \ldots, g_e$ of polynomials in $\mathbb{K}(T^1)[y], \ldots, \mathbb{K}(T^e)[y]$, the operation $\mathrm{xgcd}(\mathbf{f}, \mathbf{g}, \mathbf{T})$ returns a sequence $(((g_{ij}, u_{ij}, v_{ij}, T^{ij}), 1 \leq j \leq e_i), 1 \leq i \leq e)$ where $((g_{ij}, u_{ij}, v_{ij}, T^{ij}), 1 \leq j \leq e_i)$ is an extended greatest common divisor of f_i and g_i and such that $(T^{ij}, 1 \leq j \leq e_i, 1 \leq i \leq e)$ is a non-critical refinement of $\mathbf{T}$.

Proposition 4.1 implies that if $f_1, \ldots, f_e, g_1, \ldots, g_e$ have degree at most d then $\mathrm{xgcd}(\mathbf{f}, \mathbf{g}, \mathbf{T})$ runs in at most $O(\mathsf{M}(d)\log(d))\mathsf{A}_n(T)$ operations in k.

5 Fast computation of quasi-inverses

Throughout this section, we consider an arithmetic time A_{n-1} for triangular sets in $n-1$ variables. We explain how a quasi-inverse can be computed fast with the algorithms *split*, *xgcd*, and *removeCriticalPairs*.

Proposition 5.1. *Let $T = (T_1, \ldots, T_n)$ be a triangular set with $\deg_i(T) = d_i$ for all $1 \leq i \leq n$. Let f be in $\mathbb{K}(T)$. Then one can compute a quasi-inverse of f modulo T in $O\big(\mathsf{M}(d_n)\log(d_n)\big)\mathsf{A}_{n-1}(T_{<n})$ operations in k.*

We consider first the case where f is a non-constant polynomial and its degree w.r.t. X_n is positive and less than d_n; we give the algorithm, followed by the necessary explanations. Here, the quantity at the end a line, once multiplied by $\mathsf{A}_{n-1}(T_{<n})$, gives the total amount of time spent at this line. At the end of this section, we briefly discuss the other cases to be considered for f.

quasiInverse$_n(f, T)$ ==

1	$((g_i, u_i, v_i, T^i_{<n}), 1 \le i \le e) := \text{xgcd}(f, T_n, T_{<n})$	$\big[O\big(\mathsf{M}(d_n)\log(d_n)\big)\big]$
2	$(T^i_n, \ldots, T^e_n) := \text{split}(T_n, \{T_{<n}\}, \{T^1_{<n}, \ldots, T^e_{<n}\})$	$[O(d_n)]$
3	$(f_i, \ldots, f_e) := \text{split}(f, \{T_{<n}\}, \{T^1_{<n}, \ldots, T^e_{<n}\})$	$[O(d_n)]$
4	$\mathbf{T} := \{\}$; $\mathbf{C} := \{\}$; **result** $:= \{\}$;	
5	**for** $i = 1 \ldots e$ **do**	
5.1	**if** $\deg(g_i) = 0$ **then**	
5.1.1	$\mathbf{C} := \mathbf{C},\ (u_i, T^i_{<n} \cup T^i_n)$; $\mathbf{T} := \mathbf{T},\ T^i_{<n} \cup T^i_n$	
5.2	**else if** $\deg(g_i) > 0$ **then**	
5.2.1	$\mathbf{C} := \mathbf{C},\ (0, T^i_{<n} \cup g_i)$; $\mathbf{T} := \mathbf{T},\ T^i_{<n} \cup g_i$	
5.2.2	$q_i := \text{quotient}(T^i_n, g_i)$	$[5\mathsf{M}(d_n) + O(d_n)]$
5.2.3	$((g_{ij}, u_{ij}, v_{ij}, T^{ij}_{<n}), 1 \le j \le e_i) := \text{xgcd}(f_i, q_i, T^i_{<n})$	
5.2.4	$(T^{i1}_n, \ldots, T^{ie_i}_n) := \text{split}(q_i, \{T^i_{<n}\}, \{T^{i1}_{<n}, \ldots, T^{ie_i}_{<n}\})$	$[O(d_n)]$
5.2.5	**for** $j = 1 \ldots e_i$ **do**	
5.2.5.1	$\mathbf{C} := \mathbf{C},\ (u_{ij}, T^{ij}_{<n} \cup T^{ij}_n)$; $\mathbf{T} := \mathbf{T},\ T^{ij}_{<n} \cup T^{ij}_n$	
6	$\mathbf{T'_{<n}} := \text{removeCriticalPairs}(\mathbf{T_{<n}})$	$[O(1)]$
7	**for** $(u, S) \in \mathbf{C}$ **do**	
7.1	$(R^1, \ldots, R^l) := \text{decomp}(S_{<n}, \mathbf{T'_{<n}})$	
7.2	$(S^1_n, \ldots, S^l_n) := \text{split}(S_n, \{S_{<n}\}, \{R^1, \ldots, R^l\})$	$[O(d_n)]$
7.3	$(u_1, \ldots, u_l) := \text{split}(u, \{S_{<n}\}, \{R^1, \ldots, R^l\})$	$[O(d_n)]$
7.4	**result** $:=$ **result**, $((u_k, R^k \cup S^k_n), 1 \le k \le l)$	
8	**return result**	

We first calculate an extended greatest common divisor of f and T_n modulo the triangular set $T_{<n} = (T_1, \ldots, T_{n-1})$. This induces a non-critical decomposition $\{T^1_{<n}, \ldots, T^e_{<n}\}$ of $T_{<n}$. For further operations, we compute the images of T_n and f over this decomposition.

Let $1 \le i \le e$. If the value of g_i is 1, then u_i is the inverse of f modulo $\{T^i_{<n} \cup T^i_n\}$. Otherwise, $\deg g_i > 0$, and the computation needs to be split into two branches.

In one branch, at line 5.2.1, we build the triangular set $\{T^i_{<n} \cup g_i\}$, modulo which f reduces to zero. In the other branch, starting from line 5.2.2, we build the triangular set as $\{T^i_{<n} \cup q_i\}$, modulo which f is invertible. Indeed since the triangular set $\{T^i_{<n} \cup q_i\}$ generates a radical ideal, T^i_n is squarefree modulo $\{T^i_{<n}\}$, and $\gcd(f, q_i)$ must be 1 modulo $\{T^i_{<n} \cup q_i\}$. Therefore we can simply use the *xgcd* (step 5.2.3) once to compute the quasi-inverse of f modulo $\{T^i_{<n} \cup q_i\}$.

After collecting all the quasi-inverses, we remove the critical pairs in the new family of

triangular sets. Since no critical pairs are created at level n in the previous computation, the removal of critical pairs needs only to perform below level n. At the end, we split the inverses and the top polynomials w.r.t the last non-critical decomposition.

We also need quasi-inverse computations in two other different situations. One is when f may not have the same main variable as the triangular set T. We need also to compute the quasi-inverses in the sense of quasiInverse($\mathbf{f}, \mathbf{T}$) introduced in Section 4 where $\mathbf{T} = T^1, \ldots, T^e$ is a triangular decomposition of T, and $\mathbf{f} = f_1, \ldots, f_e$ is a sequence of polynomials in $k[X_1, \ldots, X_n]$. They are simply built on top of the quasiInverse$_n(f, T)$, with additional splits and removal of critical pairs. The dominant cost is the two xgcd calls. Therefore, in each situation, the total cost is bounded by $O\big(\mathsf{M}(d_n)\log(d_n)\big)\mathsf{A}_{n-1}(T_{<n})$.

6 Coprime factorization

We present in this section a quasi-linear time algorithm for coprime factorization of univariate polynomials over a field. Other fast algorithms for this problem are given by (Gautier and Roch, 1997), with a concern for parallel efficiency, and in (Bernstein, 2005), in a wider setting, but with a slightly worse computation time. Remark that the research announcement (Bernstein, 2004) has a time complexity that essentially matches ours.

Due to space consideration, we present our algorithm only for polynomials over a field k; however, it adapts over a direct product of fields, following the ideas presented in Section 4. We will use this tool in Section 7 for computing non-critical refinements of a triangular decomposition (see the example in the introduction for a motivation of this idea).

Definition 6.1. Let $A = a_1, \ldots, a_s$ be squarefree polynomials in $k[x]$. Some polynomials $b_1, \ldots, b_t$ in $k[x]$ are a *gcd-free basis* of the set A if $\gcd(b_i, b_j) = 1$ for $i \neq j$, each a_i can be written (necessarily uniquely) as a product of some of the b_j, and each b_j divides one of the a_i. The associated *coprime factorization* of A consists in the factorization of all polynomials a_i in terms of the polynomials $b_1, \ldots, b_t$.

Proposition 6.2. *Let d be the sum of the degrees of $A = a_1, \ldots, a_s$. Then a coprime factorization of A can be computed in $O(\mathsf{M}(d)\operatorname{logp}(d)^3)$ operations in k.*

For brevity's sake, we will only prove how to compute a gcd-free basis of A, assuming without loss of generality that all a_i have positive degree. Deducing the coprime factorization of A involves some additional bookkeeping operations, keeping track of divisibility relations; it induces no new arithmetic operations, and thus has no consequence on complexity.

The subproduct tree. The subproduct tree is a useful construction to devise fast algorithms with univariate polynomials, in particular the coprime factorization. We review this notion briefly and refer to (von zur Gathen and Gerhard, 1999) for more details.

Let $m_1, \ldots, m_r$ be monic, non-constant, polynomials in $k[x]$. The subproduct tree Sub associated to $m_1, \ldots, m_r$ is defined as follows. If $r = 1$, then Sub is a single node, labeled by the polynomial m_1. Else, let $r' = \lceil r/2 \rceil$, and let Sub_1 and Sub_2 be the trees associated

to $m_1, \ldots, m_{r'}$ and $m_{r'+1}, \ldots, m_r$ respectively. Let p_1 and p_2 be the polynomials at the roots of Sub_1 and Sub_2. Then Sub is the tree whose root is labeled by the product $p_1 p_2$ and has children Sub_1 and Sub_2. A *row* of the tree consists in all nodes lying at some given distance from the root. The *depth* of the tree is the number of its non-empty rows. Let $d = \sum_{i=1}^{r} \deg(m_i)$; then the sum of the degrees of the polynomials on any row of the tree is at most d, and its depth is at most $\operatorname{logp}(d)$.

We now define some subroutines required for our gcd-free basis algorithm, starting by the computation of multiple gcd's. Recall that the cost at given any line in our pseudo-code denotes the total time spent at this line; for simplicity, in what follows, we omit the $O(\ )$ in the complexity estimates attached to the pseudo-code.

Multiple gcd's. The first algorithm takes as input p and $(a_1, \ldots, a_e)$ in $k[x]$, and outputs the sequence of all $\gcd(p, a_i)$. The idea of this algorithm is to first reduce p modulo all a_i using fast simultaneous reduction, and then take the gcd's of all remainders with the polynomials a_i (see also Exercise 11.4 in (von zur Gathen and Gerhard, 1999)).

We make the assumption that all a_i are non-constant in the pseudo-code below, so as to apply the results of Proposition 2.2. To cover the general case, it suffices to introduce a wrapper function, that strips the input sequence $(a_1, \ldots, a_e)$ from its constant entries, and produces 1 as corresponding gcd's; this function induces no additional arithmetic cost. Finally, we write $d = \sum_{i=1}^{e} \deg a_i$.

multiGcd$(p, (a_1, \ldots, a_e))$ ==

1	**if** $\deg p \geq d$ **then** $p := p \bmod (a_1 \cdots a_e)$	$[\mathsf{M}(\deg p) + \mathsf{M}(d)\operatorname{logp}(d)]$
2	$(q_1, \ldots, q_e) := (p \bmod a_1, \ldots, p \bmod a_e)$	$[\mathsf{M}(d)\operatorname{logp}(d)]$
3	**return** $(\gcd(q_1, a_1), \ldots, \gcd(q_e, a_e))$	$\big[\sum_i \mathsf{M}(\deg a_i)\operatorname{logp}(\deg a_i)\big]$

The cost of lines 1 and 2 follows from Proposition 2.2. The function $d \mapsto \mathsf{M}(d)\operatorname{logp}(d)$ is super-additive, so the complexity at line 3 fits in $O(\mathsf{M}(d)\operatorname{logp}(d))$. Hence, the total cost of this algorithm is in $O(\mathsf{M}(\deg p) + \mathsf{M}(d)\operatorname{logp}(d))$.

Pairs of gcd's. The next step is to compute several pairs of gcd's. On input, we take two families of polynomials $(a_1, \ldots, a_e)$ and $(b_1, \ldots, b_s)$, where all a_i (resp. all b_i) are squarefree and pairwise coprime. The following algorithm computes all $\gcd(a_i, b_j)$. As above, we suppose that all a_i are non-constant; to cover the general case, it suffices to introduce a wrapper function, with arithmetic cost 0, that removes each constant a_i from the input, and adds the appropriate sequence $(1, \ldots, 1)$ in the output. Here, we write $d = \max(\sum_i \deg a_i, \sum_j \deg b_j)$.

pairsOfGcd$((a_1, \ldots, a_e), (b_1, \ldots, b_s))$ ==

1	Build a subproduct tree $\mathsf{Sub}(a_1, \ldots, a_e)$ and let $f = \text{RootOf}(\mathsf{Sub})$	$[\mathsf{M}(d)\operatorname{logp}(d)]$
2	Label the root of Sub by multiGcd$(f, \{b_1, \ldots, b_s\})$	$[\mathsf{M}(d)\operatorname{logp}(d)]$
3	**for** every node $N \in \mathsf{Sub}$, going top-down **do**	
3.1	**if** N is not a leaf and has label **g then**	

3.1.1 $f_1 := \text{leftChild}(N)$; $f_2 := \text{rightChild}(N)$;
3.1.2 Label f_1 by $\text{multiGcd}(f_1, \mathbf{g})$ $[\mathsf{M}(d)\log\mathrm{p}(d)^2]$
3.1.3 Label f_2 by $\text{multiGcd}(f_2, \mathbf{g})$ $[\mathsf{M}(d)\log\mathrm{p}(d)^2]$

This algorithm computes the gcd's of $(b_1, \ldots, b_s)$ with all polynomials in the subproduct tree associated with $(a_1, \ldots, a_e)$; the requested output can be found at the leaves of the tree. To give the complexity of this algorithm, one proves that the total number of operations along each row is in $O(\mathsf{M}(d)\log\mathrm{p}(d))$, whence a total cost in $O(\mathsf{M}(d)\log\mathrm{p}(d)^2)$.

A special case of gcd-free basis. The input of our third subroutine are sequences of polynomials $(a_1, \ldots, a_e)$ and $(b_1, \ldots, b_s)$, where all a_i (resp. all b_i) are squarefree and pairwise coprime. We compute a gcd-free basis of $(a_1, \ldots, a_e, b_1, \ldots, b_s)$; this is done by computing all $\gcd(a_i, b_j)$, as well as the quotients $\delta_i = a_i / \prod_j \gcd(a_i, b_j)$ and $\gamma_j = b_j / \prod_i \gcd(a_i, b_j)$.

We denote by $\text{removeConstants}(L)$ a subroutine that removes all constant polynomials from a sequence L (such a function requires no arithmetic operation, so its cost is zero in our model). In the complexity analysis, we still write $d = \max(\sum_i \deg a_i, \sum_j \deg b_j)$.

$\text{gcdFreeBasisSpecialCase}((a_1, \ldots, a_e), (b_1, \ldots, b_s)) ==$
1 $(g_{i,j})_{1\le i\le e, 1\le j\le s} := \text{pairsOfGcd}((a_1, \ldots, a_e), (b_1, \ldots, b_s))$ $[\mathsf{M}(d)\log\mathrm{p}(d)^2]$
2 **for** j **in** $1 \ldots s$ **do**
2.1 $L_j := \text{removeConstants}(g_{1,j}, \ldots, g_{e,j})$
2.2 $\beta_j := \prod_{\ell \in L_j} \ell$ $[\mathsf{M}(d)\log\mathrm{p}(d)]$
2.3 $\gamma_j := b_j \text{ quo } \beta_j$ $[\mathsf{M}(d)]$
3 **for** i **in** $1 \ldots e$ **do**
3.1 $L_i := \text{removeConstants}(g_{i,1}, \ldots, g_{i,s})$
3.2 $\alpha_i := \prod_{\ell \in L_i} \ell$ $[\mathsf{M}(d)\log\mathrm{p}(d)]$
3.3 $\delta_i := a_i \text{ quo } \alpha_i$ $[\mathsf{M}(d)]$
4 **return** $\text{removeConstants}(g_{1,1}, \ldots, g_{i,j}, \ldots, g_{e,s}, \gamma_1, \ldots, \gamma_s, \delta_1, \ldots, \delta_e)$

The validity of this algorithm is easily checked. The estimates for the cost of lines 2.2, 2.3, 3.2 and 3.3 come for the cost necessary to build a subproduct tree and perform Euclidean division, together with the fact that β_j (resp. α_i) divides b_j (resp. a_i). The total cost is thus in $O(\mathsf{M}(d)\log\mathrm{p}(d)^2)$.

Gcd-free basis. We finally give an algorithm for gcd-free basis. As input, we take squarefree, non-constant polynomials $a_1, \ldots, a_e$, with $d = \sum_{i\le e} \deg a_i$. We need a construction close to the subproduct tree: we form a binary tree Sub' whose nodes will be labeled by sequences of polynomials. Initially the leaves contain the sequences of length 1 $(a_1), \ldots, (a_e)$, and all other nodes are empty. Then, we go up the tree; at a node N, we use the subroutine above to compute a gcd-free basis of the sequences labeling the children of N.

$\text{gcdFreeBasis}(\{a_1, \ldots, a_e\}) ==$
1 Build the tree $\mathsf{Sub}'(a_1, \ldots, a_e)$
2 **for** every node $N \in \mathsf{Sub}'$ **and** from bottom-up **repeat**

2.1 **if** N is not a leaf **then**
2.1.1 $f_1 := \text{leftChild}(N)$; $f_2 := \text{rightChild}(N)$
2.1.2 Label N by gcdFreeBasisSpecialCase(f_1, f_2) $[\mathsf{M}(d)\log\mathrm{p}(d)^3]$
3 **return** the label of RootOf(Sub')

The total number of operations at a node N of the subset tree is $O(\mathsf{M}(d_N)\log\mathrm{p}(d_N)^2)$, where d_N is sum of the degrees of the polynomials lying at the two children of N. Summing over all nodes, using the tree structure, the total cost is seen to be in $O(\mathsf{M}(d)\log\mathrm{p}(d)^3)$ operations, as claimed.

7 Removing critical pairs

We next show how to remove critical pairs. This is an inductive process, whose complexity is estimated in the following proposition and its corollary. We need to extend the notion of "refining" introduced previously. Extending Definition 1.3, we say that a family of triangular sets $\mathbf{T}'$ refines another family $\mathbf{T}$ if for every $T \in \mathbf{T}$, there exists a subset of $\mathbf{T}'$ that forms a triangular decomposition of $\langle T\rangle$. Note the difference with the initial definition: we do not impose that the family $\mathbf{T}$ forms a triangular decomposition of some ideal I. In particular, the triangular sets in $\mathbf{T}$ do not have to generate coprime ideals.

Proposition 7.1. *There exists a constant K such that the following holds. Let $\mathsf{A}_1, \ldots, \mathsf{A}_{n-1}$ be arithmetic times for triangular sets in $1, \ldots, n-1$ variables.*

Let T be a triangular set in n variables, and let $\mathbf{U}$ be a triangular decomposition of $\langle T\rangle$. Then for all $j = 1, \ldots, n$, the following holds: given $\mathbf{U}_{\leq j}$, one can compute a non-critical triangular decomposition $\mathbf{W}$ of $T_{\leq j}$ that refines $\mathbf{U}_{\leq j}$ using a_j operations in k, where a_j satisfies the recurrence inequalities $a_0 = 0$ and for $j = 0, \ldots, n-1$,

$$a_{j+1} \leq 2a_j + K\mathsf{M}(d_{j+1}\cdots d_n)\log\mathrm{p}(d_{j+1}\cdots d_n)^3\mathsf{A}_j(T_{\leq j}),$$

and where $d_j = \deg_j T$ for $j = 1, \ldots, n$.

Before discussing the proof of this assertion, let us give an immediate corollary, which follows by a direct induction.

Corollary 7.2. *Given a triangular decomposition $\mathbf{U}$ of $\langle T\rangle$, one can compute a non-critical triangular decomposition $\mathbf{W}$ of $\langle T\rangle$ that refines $\mathbf{U}$ in time*

$$K\left(2^{n-1}\mathsf{M}(d_1\cdots d_n)\log\mathrm{p}(d_1\cdots d_n)^3 + \cdots + \mathsf{M}(d_n)\log\mathrm{p}(d_n)^3\mathsf{A}_{n-1}(T_{\leq n-1})\right).$$

PROOF. We only sketch the proof of the proposition. Let thus j be in $0, \ldots, n-1$ and let $\mathbf{U} = U^1, \ldots, U^e$ be a triangular decomposition of $\langle T\rangle$; we aim at removing the critical pairs in $\mathbf{U}_{\leq j+1}$. Let $\mathbf{V}$ be obtained by removing the critical pairs in $\mathbf{U}_{\leq j}$. Thus, $\mathbf{V}$ consists in triangular sets in $k[X_1, \ldots, X_j]$, and has no critical pair.

Let us fix $i \leq e$, and write $U^i = (U_1^i, \ldots, U_n^i)$. By definition, there exists a subset $\mathbf{V}_i = V^{i,1}, \ldots, V^{i,e_i}$ of $\mathbf{V}$ which forms a non-critical decomposition of $(U_1^i, \ldots, U_j^i)$. Our next step is to compute

$$U_{j+1}^{i,1}, \ldots, U_{j+1}^{i,e_i} = \mathrm{split}(U_{j+1}^i, (U_1^i, \ldots, U_j^i), \mathbf{V}_i).$$

Consider now a triangular set V in $\mathbf{V}$. There may be several subsets $\mathbf{V}_i$ such that $V \in \mathbf{V}_i$. Let $S_V \subset \{1, \ldots, e\}$ be the set of corresponding indices; thus, for any $i \in S_V$, there exists $\ell(i)$ in $1, \ldots, e_i$ such that $V = V^{i,e_{\ell(i)}}$. We will then compute a coprime factorization of all polynomials $U_{j+1}^{i,e_{\ell(i)}}$ in $\mathbb{K}(V)[X_{j+1}]$, for $i \in S_V$, and for all V.

This process will refine the family $\mathbf{V}$, creating possibly new critical pairs: we get rid of these critical pairs, obtaining a decomposition $\mathbf{W}$. It finally suffices to split all polynomials in the coprime factorization obtained before from $\mathbf{V}$ to $\mathbf{W}$ to conclude. The cost estimates then takes into account the cost for the two calls to the same process in j variables, hence the term $2a_j$, and the cost for coprime factorization and splitting. Studying the degrees of the polynomials involved, this cost can be bounded by

$$K\mathsf{M}(d_{j+1} \cdots d_n) \log\mathrm{p}(d_{j+1} \cdots d_n)^3 \mathsf{A}_j(T_{\leq j})$$

for some constant K, according to the results in the last section. □

8 Concluding the proof

All ingredients are now present to give the proof of the following result, which readily implies the main theorems stated in the introduction.

Theorem 8.1. *There exists a constant L such that, writing*

$$\mathsf{A}_n(d_1, \ldots, d_n) = L^n \prod_{i \leq n} \mathsf{M}(d_i) \log\mathrm{p}(d_i)^3,$$

the function $T \mapsto \mathsf{A}_n(\deg_1 T, \ldots, \deg_n T)$ is an arithmetic time for triangular sets in n variables, for all n.

Proof. The proof requires to check that taking L big enough, all conditions defining arithmetic times are satisfied. We do it by induction on n; the case $n = 1$ is settled by Proposition 2.4, taking L larger than the constant C in that proposition, and using the fact that $\log\mathrm{p}(x) \geq 1$ for all x.

Let us now consider index n; we can thus assume that the function A_j is an arithmetic time for triangular sets in j variables, for $j = 1, \ldots, n-1$. Then, at index n, condition (E_0) makes no difficulty, using the super-additivity of the function M. Addition and multiplication (condition (E_1)) and splitting (condition (E_4)) follow from Proposition 3.1, again as soon as the condition $L \geq C$ holds. The computation of quasi-inverses (condition (E_2)) is taken care of by Proposition 5.1, using our induction assumption on A, as soon as L is large enough to compensate the constant factor hidden in the $O(\ )$ estimate of that proposition.

The cost for removing critical pairs is given in the previous section. In view of Corollary 7.2, and using the condition $\mathsf{M}(dd') \leq \mathsf{M}(d)\mathsf{M}(d')$, after a few simplifications, to satisfy condition (E_3), L must satisfy the inequality

$$K(2^{n-1} + 2^{n-2}L + \cdots + L^{n-1}) \leq L^n,$$

where K is the constant introduced in Corollary 7.2. This is the case for $L \geq K + 2$. □

References

Aubry, P., Valibouze, A., 2000. Using Galois ideals for computing relative resolvents. Journal of Symbolic Computation 30 (6), 635–651.

Bernstein, D. J., 2005. Factoring into coprimes in essentially linear time. Journal of Algorithms 54 (1), 1–30.

Bernstein, D. J., 2004. D. J. Bernstein. Faster factorization into coprimes. http://cr.yp.to/papers.html#dcba2

Boulier, F., Lemaire, F., Moreno Maza, M., 2006. Well known theorems on triangular systems and the D5 Principle. In: TC'2006. University of Granada, Spain, pp. 73–86.

Brent, R., Gustavson, F., Yun, D., 1980. Fast solution of Toeplitz systems of equations and computations of Padé approximants. Journal of Algorithms 1, 259–295.

Cantor, D., Kaltofen, E., 1991. On fast multiplication of polynomials over arbitrary algebras. Acta Informatica 28, 693–701.

Dahan, X., Moreno Maza, M., Schost, É., Wu, W., Xie, Y., 2005. Lifting techniques for triangular decomposition. In: ISSAC'05. ACM press, pp. 108–115.

Dahan, X., Schost, É., 2004. Sharp estimates for triangular sets. In: ISSAC'04. ACM Press, pp. 103–110.

Della Dora, J., Dicrescenzo, C., Duval, D., 1985. About a new method method for computing in algebraic number fields. In EUROCAL 85 Vol. 2. Vol. 204 of *LNCS*. Springer-Verlag.

Dellière, S., 1999. Triangularisation de systèmes constructibles. Application à l'évaluation dynamique. Ph.D. Thesis, Université de Limoges.

Duval, D., 1994. Algebraic numbers: an example of dynamic evaluation. Journal of Symbolic Computation 18 (5), 429–446.

von zur Gathen, J., Gerhard, J., 1999. Modern Computer Algebra. Cambridge University Press.

Gautier, T., Roch, J.-L., 1997. NC2 computation of gcd-free basis and application to parallel algebraic numbers computation. In: PASCO '97. ACM Press, pp. 31–37.

Gómez Díaz, T., 1994. Quelques applications de l'évaluation dynamique. Ph.D. Thesis, Université de Limoges.

González-López, M., Recio, T., 1993. The ROMIN inverse geometric model and the dynamic evaluation method. In: Cohen, A. M. (Ed.), Proc. of the 1991 SCAFI Seminar, Computer Algebra in Industry. Wiley.

Kalkbrener, M., 1993. A generalized Euclidean algorithm for computing triangular representations of algebraic varieties. Journal of Symbolic Computation 15 (2), 143–167.

Langemyr, L., 1991. Algorithms for a multiple algebraic extension. In: Effective methods in algebraic geometry (Castiglioncello, 1990). Birkhäuser Boston, pp. 235–248.

Lazard, D., 1992. Solving zero-dimensional algebraic systems. Journal of Symbolic Computation 13 (2), 117–132.

Lombardi, H., 2006. Structures algébriques dynamiques, espaces topologiques sans points et programme de Hilbert, Annals of Pure and Applied Logic 137, 256–290.

Mora, T., 2003. Solving Polynomial Equation Systems I. The Kronecker-Duval Philosophy. No. 88 in Encyclopedia of Mathematics and its Applications. Cambridge University Press.

Moreno Maza, M., 2000. On triangular decompositions of algebraic varieties. Tech. Rep. 4/99, NAG, UK, Presented at the MEGA-2000 Conference, Bath, UK, http://www.csd.uwo.ca/∼moreno.

Moreno Maza, M., Rioboo, R., 1995. Polynomial gcd computations over towers of algebraic extensions. In: AAECC-11. Vol. 948 of *LNCS*, Springer-Verlag, pp. 365–382.

Yap, C., 1993. Fundamental Problems in Algorithmic Algebra. Princeton University Press.

Xavier Dahan
LIX, École polytechnique 91128 Palaiseau, France
dahan@lix.polytechnique.fr

Marc Moreno Maza
ORCCA, University of Western Ontario (UWO) London, Ontario, Canada
moreno@orcca.on.ca

Éric Schost
LIX, École polytechnique 91128 Palaiseau, France
schost@lix.polytechnique.fr

Yuzhen Xie
ORCCA, University of Western Ontario (UWO) London, Ontario, Canada
yxie@orcca.on.ca

L'algèbre de décomposition universelle (Universal Decomposition Algebra)

Gema M. Diaz-Toca Henri Lombardi Claude Quitté

Abstract

In this paper we present important properties of the Universal Decomposition Algebra of a polynomial over a commutative ring. Moreover, when the base ring is a field, we introduce new algorithms which make it possible to approach both splitting field of f and Galois group in a dynamic way without applying factorization algorithms.

Résumé

On donne les principales propriétés de l'algèbre de décomposition universelle d'un polynôme sur un anneau commutatif. Dans le cas d'un corps on applique ces résultats pour un traitement constructif et dynamique du corps des racines et du groupe de Galois d'un polynôme.

Introduction

Toutes les algèbres qu'on considère sont associatives, commutatives et avec élément neutre. Il revient donc au même de se donner une $\mathbf{A}$-algèbre $\mathbf{C}$ ou un homomorphisme $\mathbf{A} \xrightarrow{\rho} \mathbf{C}$.

Dans tout cet article, $\mathbf{A}$ *est un anneau commutatif,* $f = T^n + \sum_{k=1}^{n} (-1)^k a_k T^{n-k} \in \mathbf{A}[T]$ *et* $\mathbf{B} = \mathrm{Adu}_{\mathbf{A},f}$ *est l'algèbre de décomposition universelle de* f *sur* $\mathbf{A}$.

Dans la section 1 nous introduisons les modules de Cauchy et la base classique correspondante de l'algèbre de décomposition universelle. Nous montrons que les deux définitions naturelles de la norme coïncident.

Dans la section 2 nous introduisons les notions d'idempotent galoisien, d'algèbre prégaloisienne, de quotient de Galois d'une algèbre prégaloisienne. Une algèbre prégaloisienne est une algèbre qui vérifie un bon nombre de propriétés des algèbres galoisiennes, sans la condition de séparabilité. Le prototype d'une algèbre prégaloisienne est une algèbre de décomposition universelle, ou un quotient de Galois d'une algèbre de décomposition universelle.

Dans la section 3 nous montrons comment calculer des éléments galoisiens dans une algèbre de Boole munie d'un groupe d'automorphismes. Ceci s'applique en particulier à l'algèbre de Boole des idempotents d'une algèbre de décomposition universelle ou plus généralement d'une algèbre prégaloisienne.

Dans la section 4 nous améliorons les résultats connus concernant les points fixes d'une algèbre de décomposition universelle (sous l'action de S_n).

Dans la section 5 nous montrons constructivement quelques propriétés importantes qui apparaissent sous l'hypothèse de séparabilité du polynôme servant à construire l'algèbre de décomposition universelle. Nous montrons que l'algèbre de décomposition universelle est alors réduite si l'anneau de base est réduit, et plus généralement que le nilradical de $\mathbf{B}$ est engendré par celui de $\mathbf{A}$. Nous donnons une généralisation du résultat qui affirme que « l'algèbre de décomposition universelle se diagonalise elle-même lorsque le polynôme est séparable » au cas d'un quotient de Galois.

Dans la section 6 nous généralisons un résultat donné séparément par P. Aubry et A. Valibouze ([1]) et par L. Ducos ([8]) sur la structure « triangulaire » des idéaux galoisiens.

Dans la section 7, pour étudier constructivement le « corps des racines » d'un polynôme $f \in \mathbf{K}[T]$ (où $\mathbf{K}$ est un corps discret), nous proposons d'utiliser des « approximations » de ce corps qui sont des quotients convenables de l'algèbre de décomposition universelle associée à f. Dans l'article [6] le premier auteur a donné dans le même esprit un traitement de la théorie de Galois d'un polynôme séparable sur un corps discret en calcul formel. Cette étude du corps des racines « par approximations successives » peut être considérée comme une variante du système D5 [4] de traitement de la cloture algébrique en Calcul formel, ou encore comme la version constructive de l'approche de Bourbaki dans [3].

Puisque l'article est écrit dans le style des mathématiques constructives à la Bishop ([2, 10]) tous les théorèmes ont un contenu algorithmique. La terminologie constructive spécifique non précisée se trouve dans [10]. Rappelons qu'en mathématiques constructives un ensemble est dit *discret* lorsqu'on dispose d'un test d'égalité pour ses éléments.

Remerciements. Nous remercions Thierry Coquand pour ses conseils judicieux.

1 Modules de Cauchy et base canonique

On note $\mathbf{B} = \mathrm{Adu}_{\mathbf{A},f}$ l'*algèbre de décomposition universelle de f sur* $\mathbf{A}$ définie comme suit:

$$\mathbf{B} = \mathrm{Adu}_{\mathbf{A},f} = \mathbf{A}[X_1, \ldots, X_n]/\mathcal{J}(f) = \mathbf{A}[x_1, \ldots, x_n]$$

où $\mathcal{J}(f)$ est l'idéal donné par les fonctions symétriques élémentaires des X_i:

$$\alpha_1 = \sum_{i=1}^{n} X_i,\ \alpha_2 = \sum_{1\le i<j\le n} X_iX_j,\ \ldots,\ \alpha_n = \prod_{i=1}^{n} X_i\,,$$

$$\mathcal{J}(f) = \langle a_1 - \alpha_1,\ a_2 - \alpha_2, \ldots,\ a_n - \alpha_n \rangle\,.$$

L'algèbre de décomposition universelle peut être caractérisée par la propriété suivante.

Note 1.1. *(propriété caractéristique)*
Soit $\mathbf{C}$ une $\mathbf{A}$-algèbre pour laquelle $f(T)$ se décompose en produit de facteurs $(T - z_i)$. Alors il existe un unique homomorphisme de $\mathbf{A}$-algèbres $\mathbf{B} \to \mathbf{C}$ qui envoie les x_i sur les z_i. Ceci caractérise l'algèbre de décomposition universelle $\mathbf{B} = \mathrm{Adu}_{\mathbf{A},f}$, à isomorphisme unique près.

Le groupe S_n des permutations de $\{X_1,\dots,X_n\}$ agit sur $\mathbf{A}[X_1,\dots,X_n]$ et fixe l'idéal $\mathcal{J}(f)$, donc l'action passe au quotient et ceci définit S_n comme groupe d'automorphismes de l'algèbre de décomposition universelle.

Note 1.2. *(changement d'anneau de base)*
Soit $\rho : \mathbf{A} \to \mathbf{A}_1$ une $\mathbf{A}$-algèbre. Notons $\rho(f)$ l'image de f dans $\mathbf{A}_1[T]$. Alors l'algèbre $\mathrm{Adu}_{\mathbf{A},f} \otimes_{\mathbf{A}} \mathbf{A}_1$, est naturellement isomorphe à $\mathrm{Adu}_{\mathbf{A}_1,\rho(f)}$.

Pour étudier l'algèbre de décomposition universelle on introduit les « modules de Cauchy » qui sont les polynômes suivants:

$$\begin{array}{rcl} f_1(X_1) & = & f(X_1) \\ f_{k+1}(X_1,\dots,X_{k+1}) & = & \frac{f_k(X_1,\dots,X_{k-1},X_k)-f_k(X_1,\dots,X_{k-1},X_{k+1})}{X_k-X_{k+1}} \quad (1 \le k \le n-1) \end{array}$$

Le polynôme f_i est symétrique en les variables $X_1,\dots,X_i$, unitaire de degré $n-i+1$ en X_i. Le fait 1.1 implique que l'idéal $\mathcal{J}(f)$ est égal à l'idéal engendré par les modules de Cauchy. Donc l'algèbre de décomposition universelle est un $\mathbf{A}$-module libre de rang $n!$.

Lemme 1.3. *Le* $\mathbf{A}$*-module* $\mathbf{B}$ *est libre et une base est formée par les « monomes »* $x_1^{d_1}\cdots x_{n-1}^{d_{n-1}}$ *tels que pour* $k=1,\dots,n-1$ *on ait* $d_k \le n-k$*. Nous noterons cette base* $\mathcal{B}(f)$*.*

Lemme 1.4. *Pour tout élément* $z \in \mathbf{B}$ *on a* $\mathrm{C}_{\mathbf{B}/\mathbf{A}}(z)(T) = \mathrm{C}_{\mathrm{S}_n}(z)(T)$*. En particulier* $\mathrm{Tr}_{\mathbf{B}/\mathbf{A}}(z) = \mathrm{Tr}_{\mathrm{S}_n}(z)$ *et* $\mathrm{N}_{\mathbf{B}/\mathbf{A}}(z) = \mathrm{N}_{\mathrm{S}_n}(z)$*.*

Démonstration. Puisque le polynôme caractéristique est la norme de $T-z$ il suffit de montrer l'égalité des deux « normes »: $\mathrm{N}_{\mathbf{B}/\mathbf{A}}(z) = \prod_{\sigma\in\mathrm{S}_n}\sigma(z)$. Écrivons $\mathrm{N}_{\mathrm{S}_n}(z)$ sur la base canonique $\mathcal{B}(f)$, c'est clairement un élément de $\mathbf{A}$. Si on prend pour coefficients de f des indéterminées a_i, pour coordonnées de z sur la base $\mathcal{B}(f)$ d'autres indéterminées et pour $\mathbf{A}$ l'anneau librement engendré par ces indéterminées on voit qu'il s'agit de démontrer une identité algébrique, c'est-à-dire une égalité entre deux éléments d'un anneau de polynômes à coefficients dans $\mathbb{Z}$. Notons N pour $\mathrm{N}_{\mathbf{B}/\mathbf{A}}$. Comme $\mathrm{N}(z) = \mathrm{N}(\sigma(z))$ pour tout $\sigma \in \mathrm{S}_n$, on obtient $\mathrm{N}(\prod_{\sigma\in\mathrm{S}_n}\sigma(z)) = \mathrm{N}(z)^{n!}$. Puisque $\mathrm{N}_{\mathrm{S}_n}(z) \in \mathbf{A}$, $\mathrm{N}(\mathrm{N}_{\mathrm{S}_n}(z)) = (\mathrm{N}_{\mathrm{S}_n}(z))^{n!}$. Ainsi $\mathrm{N}(z)$ et $\mathrm{N}_{\mathrm{S}_n}(z)$ sont deux polynômes en les indéterminées qui sont égaux après avoir été élevés à la puissance $n!$. Puisqu'on est dans un anneau factoriel, on doit avoir $\mathrm{N}_{\mathrm{S}_n}(z) = c\,\mathrm{N}(z)$ avec $c \in \mathbb{Z}$ et $c^{n!} = 1$. Enfin, puisque toute situation particulière est obtenue comme spécialisation de la situation générale (avec des coefficients indéterminés), pour connaître c on peut spécialiser z en 1: $c = 1$. □

2 Idempotents galoisiens dans une algèbre prégaloisienne

Dans la suite nous notons $\mathbb{B}(\mathbf{C})$ l'algèbre de Boole des idempotents d'un anneau $\mathbf{C}$. Les opérations sont $u \wedge v := uv$, $u \vee v := u+v-uv$, $u \oplus v := u+v-2uv = (u-v)^2$, $\neg u := 1-u = 1 \oplus u$. Et la relation d'ordre partiel est $u \le v \iff u \wedge v = u \iff u \vee v = v$.

Dans une algèbre de Boole un élément non nul minimal est appelé un *atome*. Dans le cas d'un anneau on parle d'*idempotent indécomposable*.

Rappelons qu'un automorphisme σ d'un anneau $\mathbf{C}$ est dit *séparant* s'il existe $x_1,\ldots x_k$, $a_1,\ldots,a_k \in \mathbf{C}$ tels que $1 = \sum_{i=1}^{k} a_i(x_i - \sigma(x_i))$ et qu'un groupe G d'automorphismes de $\mathbf{C}$ est dit *séparant* si les éléments $\neq \mathrm{Id}_{\mathbf{C}}$ de G sont séparants. Une *algèbre galoisienne* est par définition un triplet $(\mathbf{A}, \mathbf{C}, G)$ où G est un groupe séparant d'automorphismes de $\mathbf{C}$ et $\mathbf{A} = \mathrm{Fix}(G)$ est le sous-anneau des points fixes de G (cf. [5]).

Nous utiliserons les notations suivantes lorsqu'un groupe G opère sur un ensemble E. Pour $x \in E$, $\mathrm{St}(x)$ désigne le stabilisateur de x; pour $F \subset E$, $\mathrm{Stp}(F)$ désigne le stabilisateur point par point de F et $\mathrm{Stab}(F)$ le stabilisateur de F. Et si $H \subset G$, $\mathrm{Fix}(H) = E^H$ désigne la partie de E formée par les éléments fixés par tous les $\sigma \in H$.

Nous donnons maintenant une définition qui permet d'insérer l'algèbre de décomposition universelle dans un cadre un peu plus général et utile.

Définition 2.1. *(algèbre prégaloisienne)*
Une algèbre prégaloisienne est donnée par un triplet $(\mathbf{A}, \mathbf{C}, G)$ où

1. $\mathbf{C}$ est une $\mathbf{A}$-algèbre avec $\mathbf{A} \subset \mathbf{C}$, $\mathbf{A}$ facteur direct dans $\mathbf{C}$,
2. G est un groupe fini de $\mathbf{A}$-automorphismes de $\mathbf{C}$,
3. $\mathbf{C}$ est un $\mathbf{A}$-module projectif de rang constant $|G|$,
4. pour tout $z \in \mathbf{C}$, $\mathrm{C}_{\mathbf{C}/\mathbf{A}}(z) = \mathrm{C}_G(z)$.

Par exemple $(\mathbf{A}, \mathbf{B}, \mathrm{S}_n)$ est une algèbre prégaloisienne.

NB: La notion d'algèbre prégaloisienne est un peu moins contraignante que la notion plus usuelle d'algèbre galoisienne.

Définition 2.2. Dans une algèbre prégaloisienne $(\mathbf{A}, \mathbf{C}, G)$ un idempotent e de $\mathbf{C}$ est dit *galoisien* si son orbite sous G est un système fondamental d'idempotents orthogonaux (sfio). Un idéal de $\mathbf{C}$ est dit *galoisien* lorsqu'il est engendré par l'idempotent complémentaire d'un idempotent galoisien.

Théorème 2.3 (théorème de Structure 1). *Soit une algèbre prégaloisienne $(\mathbf{A}, \mathbf{C}, G)$, e un idempotent galoisien de $\mathbf{C}$, et $\{e_1, \ldots, e_m\}$ son orbite sous G. Soit H le stabilisateur de $e = e_1$ et $r = |H|$, de sorte que $rm = |G|$. Posons $\mathbf{C}_i = \mathbf{C}/\langle 1 - e_i\rangle \simeq e_i\mathbf{C}$ $(1 \leq i \leq m)$. Soit enfin $\pi : \mathbf{C} \to \mathbf{C}_1$ la projection canonique.*

1. *Les $\mathbf{C}_i$ sont des $\mathbf{A}$-algèbres deux à deux isomorphes, et $\mathbf{C} \simeq \mathbf{C}_1^m$ (comme $\mathbf{A}$-algèbres).*
2. *L'algèbre $\mathbf{C}_1$ est un $\mathbf{A}$-module projectif de rang constant $r = |H|$. La restriction de π à $\mathbf{A}$, et même à $\mathbf{C}^G$, est injective. Et $\mathbf{A}$ (identifié à son image dans $\mathbf{C}_1$) est facteur direct dans $\mathbf{C}_1$.*
3. *Le groupe H opère sur $\mathbf{C}_1$ et $\mathbf{C}_1^H$ est canoniquement isomorphe à $\mathbf{C}^G$: plus précisément $\mathbf{C}_1^H = \pi(\mathbf{C}^H) = \pi(\mathbf{C}^G)$.*
4. *Pour tout $z \in \mathbf{C}_1$, $\mathrm{C}_{\mathbf{C}_1/\mathbf{A}}(z)(T) = \mathrm{C}_H(z)(T)$.*

5. $(\mathbf{A}, \mathbf{C}_1, H)$ *est une algèbre prégaloisienne, on dira que c'est un* quotient de Galois *de* $(\mathbf{A}, \mathbf{C}, G)$.

6. *Soit* g_1 *un idempotent galoisien de* $(\mathbf{A}, \mathbf{C}_1, H)$, K *son stabilisateur dans* H, $g' \in e_1\mathbf{C}$ *tel que* $\pi(g') = g_1$. *Alors* g' *est un idempotent galoisien de* $(\mathbf{A}, \mathbf{C}, G)$, *son stabilisateur est* K, *et on a un isomorphisme canonique* $\mathbf{C}_1/\langle 1 - g_1\rangle \simeq \mathbf{C}/\langle 1 - g'\rangle$.

Démonstration. Le point 1 est évident. La première affirmation du point 2 en est une conséquence immédiate. Soit $\tau_1 = \mathrm{Id}, \tau_2, \ldots, \tau_m$ un système de représentants pour G/H, avec $\tau_i(e_1) = e_i$. Montrons que la restriction de π à $\mathbf{C}^G$ est injective: si $a \in \mathbf{C}^G$ et $e_1 a = 0$ alors en transformant par les τ_j, tous les $e_j a$ sont nuls, et donc aussi leur somme, égale à a. Montrons que $\pi(\mathbf{A})$ est facteur direct dans $\mathbf{C}_1$. Soit $\lambda : \mathbf{C}_1 \to e_1\mathbf{C}$ l'isomorphisme réciproque de la restriction de π à $e_1\mathbf{C}$. Il s'agit d'un isomorphisme de $\mathbf{A}$-algèbres, e_1 étant l'élément neutre pour la multiplication dans $e_1\mathbf{C}$. Soit $\varphi : \mathbf{C} \to \mathbf{A}$ une forme $\mathbf{A}$-linéaire vérifiant $\varphi(1) = 1$. On définit $\psi : \mathbf{C}_1 \to \pi(\mathbf{A})$ par $\psi(y) = \pi(\varphi(x + \tau_2(x) + \cdots + \tau_m(x)))$ où $x = \lambda(y)$. On a bien que ψ est une forme linéaire vérifiant $\psi(1) = 1$ donc $\mathbf{C}_1 = \pi(\mathbf{A}) \oplus \mathrm{Ker}\,\psi$.
Voyons le point 3. Montrons d'abord $\mathbf{C}_1^H = \pi(\mathbf{C}^H)$. Soit $u \in \mathbf{C}$ tel que $\pi(u) = z \in \mathbf{C}_1^H$. Puisque $z \in \mathbf{C}_1^H$, pour tout $\sigma \in H$, $\sigma(u) \equiv u \bmod \langle 1 - e_1\rangle$, ce qui signifie, $e_1\sigma(u) = e_1 u$. Comme $\sigma(e_1) = e_1$ et $\pi(e_1) = 1_{\mathbf{C}_1}$ on obtient avec $y = e_1 u$: $\pi(y) = z$ et pour tout $\sigma \in H$, $\sigma(y) = y$ c'est-à-dire $z \in \pi(\mathbf{C}^H)$.
Montrons maintenant que $z \in \pi(\mathbf{C}^G)$. On pose $v = \sum_i \tau_i(y) = \sum_i \tau_i(e_1 y) = \sum_i e_i\tau_i(y)$. On a $\pi(e_i) = \delta_{1i}$ et donc $\pi(v) = \pi(y)$. Montrons que v est fixe par G. Si $\sigma \in G$, $\sigma(v) = \sum_i \sigma(e_i\tau_i(y))$. Fixons i et posons $\sigma(e_i) = e_j$. Notre but est de montrer que $\sigma(\tau_i(y)) = \tau_j(y)$, c'est-à-dire que $\tau_j^{-1}\sigma\tau_i$ fixe y. Or cela résulte de $y \in \mathbf{C}^H$ et $\tau_j^{-1}\sigma\tau_i \in H$ puisque $\tau_j^{-1}\sigma\tau_i(e_1) = e_1$.
Voyons le point 4. Soit u tel que $\pi(u) = z$ et $y = e_1 u$. On a $\pi_i(y) = 0$ pour $i \neq 1$ et $\pi(y) = z$. Dans la décomposition $\mathbf{C} = e_1\mathbf{C} \oplus \cdots \oplus e_m\mathbf{C}$, y s'ecrit donc $(y, 0, \ldots, 0)$ et $T - y$ s'écrit $(T - y, T, \ldots, T)$. Cela donne $\mathrm{C}_{\mathbf{C}/\mathbf{A}}(y)(T) = T^p\,\mathrm{C}_{\mathbf{C}_1/\mathbf{A}}(z)$ avec $p = |G| - |H|$. En considérant $\sigma(y)$ pour un $\sigma \in G$ arbitraire on peut écrire $\sigma = \tau_i\lambda$ pour un certain i et un élément λ de H. Ceci permet de voir que la composante dans $e_1\mathbf{C}$ de $\mathrm{C}_G(y)(T)$ n'est autre que $T^p\,\mathrm{C}_H(z)(T)$ (qu'on remonte de $\mathbf{C}_1[T]$ dans $e_1\mathbf{C}[T]$). Par raison de symétrie il en sera de même pour les autres composantes, c'est-à-dire qu'on a $\mathrm{C}_G(y)(T) = T^p\,\mathrm{C}_H(z)(T)$.
Le point 5 est une synthèse des points précédents.
Voyons le point 6. En tenant compte du fait que la restriction de π à $e_1\mathbf{C}$ est un isomorphisme on a ${g'}^2 = g' = g'e_1$. De même pour $\sigma \in H$ on a: $\sigma(g') = g'$ si $\sigma \in K$, ou $g'\sigma(g') = 0$ si $\sigma \notin K$. Enfin pour $\sigma \in G \setminus H$, $e_1\sigma(e1) = 0$ et donc $g'\sigma(g') = 0$. Ceci montre que g' est un idempotent galoisien de $\mathbf{B}$ avec pour stabilisateur K. L'isomorphisme canonique est immédiat. □

3 Éléments galoisiens dans une algèbre de Boole

Le lemme suivant constitue un raffinement constructif de la théorie des algèbres de Boole finies.

Lemme 3.1. *Soit C une algèbre de Boole. Les propriétés suivantes sont équivalentes :*

1. *C est finie.*
2. *C est discrète et de type fini.*
3. *1_C est une somme finie d'atomes.*

Dans un tel cas C est isomorphe à l'algèbre de Boole $\mathcal{P}(S)$ des parties finies de l'ensemble S des atomes.

En particulier dans le contexte des anneaux commutatifs $\mathbb{B}(\mathbf{C})$ est finie si et seulement si $1_{\mathbf{C}}$ est une somme d'idempotents indécomposables orthogonaux.

Définition 3.2. Si G est un groupe fini qui opère sur une algèbre de Boole C, un élément e de C est dit *galoisien* (pour G) si son orbite sous G est un sfio: les éléments de l'orbite sont deux à deux orthogonaux et leur somme est égale à 1.

Note 3.3. Soit G un groupe fini opérant sur une algèbre de Boole C discrète, $e \neq 0$ dans C, et $\{e_1, \ldots, e_k\}$ l'orbite de e sous G. On suppose que 1 et 0 sont les seuls éléments fixés par G. Les propriétés suivantes sont équivalentes :

1. L'élément e est galoisien.
2. Pour tout $i > 1$, $e_1 e_i = 0$.
3. Pour tout $\sigma \in G$, $e\sigma(e) = e$ ou 0.
4. Pour tous $i \neq j \in \{1, \ldots, k\}$, $e_i e_j = 0$.

Les hypothèses sont vérifiées par exemple si $C = \mathbb{B}(\mathbf{B})$, $G = \mathrm{S}_n$ et $\mathbf{A}$ est connexe.

Théorème 3.4 (théorème de structure 2). *Soit G un groupe fini opérant sur une algèbre de Boole C discrète et non triviale. On suppose que 1 et 0 sont les seuls éléments fixés par G.*

1. *Pour toute famille finie d'éléments de C il existe un élément galoisien e_1 (notons $(e_1, \ldots, e_k)$ son orbite) et tel que chaque élément e de la famille initiale vérifie $e = \sum \{e_i \mid 1 \leq i \leq k,\ e_i e \neq 0\}$.*
2. *L'algèbre de Boole C ne peut avoir plus que $2^{|G|}$ éléments.*
3. *Si e et h sont des éléments galoisiens avec $e < h$ (cad $he = e$ et $h \neq e$) si E est le stabilisateur de e et H le stabilisateur de h alors $h = \sum_{\sigma \in H/E} \sigma(e)$.*
4. *C est finie si et seulement si il existe un atome e. Dans ce cas e est galoisien, l'orbite de e est l'ensemble des atomes, G opère sur cette orbite comme sur G/E, et sur C comme sur $\mathcal{P}(G/E)$[1].*

[1] Ici, pour que l'affirmation soit valide d'un point de vue constructif, $\mathcal{P}(G/E)$ dénote l'ensemble des parties *finies* de G/E.

Démonstration. Nous montrons seulement le point 1. On considère la sous-algèbre de Boole $C' \subseteq C$ engendrée par les orbites des éléments de la famille finie donnée. C' est de type fini et discrète donc finie. En conséquence ses éléments minimaux non nuls forment un ensemble fini $S = \{e_1, \ldots, e_k\}$ et C' est isomorphe à l'algèbre de Boole des parties finies de S: $C' = \{\sum_{i\in F} e_i | F \in \mathcal{P}(\{1, \ldots, k\})\}$. Clairement G opère sur C'. Pour $\sigma \in \mathrm{S}_n$, $\sigma(e_1)$ est une somme de certains e_i, mais il ne peut y avoir deux termes dans la somme, car alors en transformant l'un de ces termes par σ^{-1} on aurait un élément non nul $< e_1$ dans C'. Donc $(e_1, \ldots, e_k)$ est un sfio et e_1 est galoisien. □

Algorithme 3.5. **Calcul d'un élément galoisien et de son stabilisateur.**

Entrée : e: élément non nul d'une algèbre de Boole C; G: groupe fini d'automorphismes de C; $S = \mathrm{St}(e)$ (sous groupe stabilisateur de e).
\# On suppose que 0 et 1 sont les seuls points fixes pour l'action de G sur C.
Sortie : e_1: élément galoisien correspondant; H: le sous groupe stabilisateur de e_1.
Variables locales : h: dans C; σ: dans G; L: liste d'éléments de G.

Début
  $e_1 \leftarrow e$; $L \leftarrow [\,]$;
  pour σ **dans** G/S **faire**
\# G/S désigne un système de représentants des classes à gauche modulo S
    $h \leftarrow e_1\sigma(e)$;
    si $h \neq 0$ **alors** $e_1 \leftarrow h$; $L \leftarrow L \bullet [\sigma]$ **fin si** ;
  fin pour
  $H \leftarrow$ le sous-groupe de G formé par les α tels que: $\forall \sigma \in L$, $\alpha\sigma \in \bigcup_{\tau\in L} \tau S$.
Fin.

Sous les hypothèses du théorème 3.4 on peut calculer un élément galoisien e_1 qui engendre la même algèbre de Boole que l'orbite de e au moyen de l'algorithme 3.5. En outre on peut également calculer le stabilisateur de e_1 (la *nouvelle approximation du groupe de Galois*) « sans sortir du groupe ». On pourra penser au cas $C = \mathbb{B}(\mathbf{B})$, $G = \mathrm{S}_n$ et $\mathbf{A}$ connexe, ou plus généralement $C = \mathbb{B}(\mathbf{C})$, $(\mathbf{A}, \mathbf{C}, G)$ est une algèbre prégaloisienne et $\mathbf{A}$ est connexe.

On notera que l'élément e_1 obtenu comme résultat du calcul dépend de l'ordre dans lequel est énuméré l'ensemble fini G/S et qu'il n'y a pas d'ordre naturel (intrinsèque) sur cet ensemble.

4 Points fixes de $\mathrm{Adu}_{\mathbf{A},f}$

Lemme 4.1. *Soit* $\mathbf{C}$ *une* $\mathbf{A}$*-algèbre qui est un module projectif de rang constant* $k \geq 1$ *(par exemple une algèbre prégaloisienne ou* $\mathbf{C} = \mathbf{B}$*).*

- *Un* $x \in \mathbf{C}$ *est inversible (resp. régulier) si et seulement si* $\mathrm{N}_{\mathbf{C}/\mathbf{A}}(x)$ *est inversible (resp. régulier) dans* $\mathbf{A}$*.*

- *Un $x \in \mathbf{A}$ est inversible (resp. régulier) dans $\mathbf{C}$ si et seulement si il est inversible (resp. régulier) dans $\mathbf{A}$.*

Lemme 4.2. *Soit J le jacobien du système de n équations à n inconnues définissant l'algèbre de décomposition universelle $\mathbf{B} = \mathrm{Adu}_{\mathbf{A},f}$.*

1. *On a $J = \prod_{1\leq i<j\leq n}(x_i - x_j)$ dans $\mathbf{B}$.*
2. *On a $J^2 = \operatorname{disc} f \in \mathbf{A}$.*
3. *En particulier les propriétés suivantes sont équivalentes :*
   (a) $\operatorname{disc} f$ *est inversible (resp. régulier) dans $\mathbf{A}$.*
   (b) *J est inversible (resp. régulier) dans $\mathbf{B}$.*
   (c) *Les $x_i - x_j$ sont inversibles (resp. réguliers) dans $\mathbf{B}$.*
   (d) *$x_1 - x_2$ est inversible (resp. régulier) dans $\mathbf{B}$.*
   (e) *$\Omega_{\mathbf{B}/\mathbf{A}} = 0$ (resp. $\Omega_{\mathbf{B}/\mathbf{A}}$ est un $\mathbf{B}$-module « de torsion », i.e. annulé par un élément régulier).*

Démonstration. Le point 1 est facile par récurrence sur n.
Le point 2 est une conséquence immédiate du point 1, et on en déduit l'équivalence des points (a) à (d) dans 3, en tenant compte du lemme 4.1.
Pour le point (e) rappelons que $\Omega_{\mathbf{B}/\mathbf{A}}$ est un $\mathbf{B}$-module isomorphe au conoyau de la matrice jacobienne, ce qui implique que $\mathrm{Ann}(\Omega_{\mathbf{B}/\mathbf{A}})$ et $J\mathbf{B}$ ont même nilradical. Enfin J est régulier (resp. inversible) si et seulement si $\sqrt{\langle J\rangle}$ contient un élément régulier (resp. contient 1). □

Nous notons $\mathrm{di}(f) = \prod_{1\leq i<j\leq n}(x_i + x_j) \in \mathbf{A}$. Il est clair que $\mathrm{di}(f)$ est congru modulo 2 à $\prod_{1\leq i<j\leq n}(x_i - x_j)$ et donc $\langle 2, \mathrm{di}(f)^2\rangle = \langle 2, \mathrm{disc}(f)\rangle$.

Théorème 4.3. *Si $\mathrm{Ann}_{\mathbf{A}}(\langle 2, \mathrm{di}(f)\rangle) = 0$ et a fortiori si $\mathrm{Ann}_{\mathbf{A}}(\langle 2, \mathrm{disc}(f)\rangle) = 0$ on a $\mathrm{Fix}(\mathrm{S}_n) = \mathbf{A}$.*

Démonstration. Puisque $\langle 2, \mathrm{di}(f)^2\rangle = \langle 2, \mathrm{disc}(f)\rangle$ un élément qui annule $\langle 2, \mathrm{di}(f)\rangle$ annule a fortiori $\langle 2, \mathrm{disc}(f)\rangle$. Il suffit donc de démontrer la deuxième affirmation.
Voyons le cas où $n = 2$. Un élément $z = c + dx_1 \in \mathbf{B}$ $(c, d \in \mathbf{A})$ est invariant par S_2 si et seulement si $d(x_1 - x_2) = d(a_1 - 2x_1) = 0$ si et seulement si $da_1 = 2d = 0$.
On procède ensuite par récurrence sur n. On écrit $\mathbf{B} = \mathbf{A} \oplus E$ où E est le $\mathbf{A}$-module engendré par les éléments $\neq 1$ de la base $\mathcal{B}(f)$. On note E' le sous module de E formé par les éléments fixes sous S_n. Pour $n > 2$ on considère l'anneau $\mathbf{A}_1 = \mathbf{A}[X_1]/\langle f(X_1)\rangle = \mathbf{A}[x_1]$, le polynôme $F(T) = f_2(T, x_1) \in \mathbf{A}_1(T)$ et l'algèbre de décomposition universelle $\mathbf{B}_1 = \mathrm{Adu}_{\mathbf{A}_1,F}$, dans laquelle nous notons $x_2, \ldots, x_n$ les variables $(X_2, \ldots, X_n$ avant de passer au quotient).
On vérifie que $\mathbf{B} \simeq \mathbf{B}_1$: une simple constatation si on utilise la définition des algèbres de décomposition universelle via les modules de Cauchy. On identifie $\mathbf{B}$ et $\mathbf{B}_1$ et on écrit $\mathbf{B}_1 = \mathbf{A}_1 \oplus E'_1$ correspondant à la base $\mathcal{B}(F)$ formée par les monomes $x_2^{d_2} \cdots x_{n-1}^{d_{n-1}}$ avec $d_i < n - i$ pour chaque i. Pour passer de l'écriure d'un élément $g \in \mathbf{B}$ sur la base $\mathcal{B}(F)$ ($\mathbf{B}$

vu comme $\mathbf{A}_1$-module) à son écriture sur la base $\mathcal{B}(f)$ ($\mathbf{B}$ vu comme $\mathbf{A}$-module), il suffit d'écrire chaque coordonnée, qui est un élément de $\mathbf{A}_1$ sur la $\mathbf{A}$-base de $\mathbf{A}_1$ formée par les monomes $1, x_1, \ldots, x_1^{n-1}$.
Notons aussi que $\mathrm{di}(f) = (-1)^{n-1}F(-x_1)\mathrm{di}(F)$ par un calcul direct et passons à la récurrence proprement dite.
Nous supposons que $\mathrm{Ann}_{\mathbf{A}}(\langle 2, \mathrm{di}(f)\rangle) = 0$. On en déduit que $\mathrm{Ann}_{\mathbf{A}_1}(\langle 2, \mathrm{di}(F)\rangle) = 0$, car si $b = \beta_0 + \beta_1 x_1 + \cdots + \beta_{n-1}x_1^{n-1} \in \mathbf{A}_1$ annule $\mathrm{di}(F)$, il annule $\mathrm{di}(f) = (-1)^{n-1}F(-x_1)\mathrm{di}(F)$, donc chacun des β_i annule $\mathrm{di}(f)$. De même chacun des β_i annule 2. Donc $b = 0$.
Soit alors $y \in E'$, écrivons $y = g(x_2, \ldots x_n)$ avec $g \in \mathbf{A}[X_2, \ldots, X_{n-1}]$ et $\deg_{X_i} g \leq n - i$ pour $i = 2, \ldots, n-1$. Autrement dit nous voyons y comme un élément de $\mathrm{Adu}_{\mathbf{A}_1,F}$. Puisque y est invariant par S_{n-1}, on en déduit par hypothèse de récurrence que $g \in \mathbf{A}_1$, c'est une « constante » qu'on écrit $h(x_1)$ avec $\deg(h) < n$. Il reste à voir que $g \in \mathbf{A}$. Si $h(X) = c_0 + c_1 X + \cdots + c_{n-1}X^{n-1}$ on écrit $h(x_1) = h(x_2)$. On note que $h(x_1)$ est l'écriture réduite de g sur la base canonique $\mathcal{B}(f)$. Concernant $h(x_2)$, pour obtenir l'écriture réduite, nous devons remplacer dans le terme $c_{n-1}x_2^{n-1}$, x_2^{n-1} par son écriture sur la base canonique, qui résulte de $f_2(x_1, x_2) = 0$. Cette réécriture fait apparaître le terme $-c_{n-1}x_1^{n-2}x_2$, et ceci implique (par l'égalité des écritures $h(x_1)$ et $h(x_2)$ sur la base $\mathcal{B}(f)$) que $c_{n-1} = 0$. Mais alors $h(x_2)$ est une écriture réduite et donc tous les c_i pour $i > 0$ sont nuls. □

Note 4.4. Le cas $\mathrm{disc}\, f$ régulier est bien connu. On le trouve avec une preuve voisine de celle ci-dessus dans la thèse de Lionel Ducos [7]. Par ailleurs Ekedahl et Laskov ont traité le cas où 2 est régulier dans [9]. Dans le cas $n = 2$ l'étude faite ci-dessus montre que dès que $\mathrm{Ann}_{\mathbf{A}}(\langle 2, \mathrm{di}\, f\rangle) \neq 0$, $\mathrm{Fix}(\mathrm{S}_2) = \mathbf{A} \oplus \mathrm{Ann}_{\mathbf{A}}(\langle 2, \mathrm{di}\, f\rangle)\, x_1$ contient strictement $\mathbf{A}$. Un calcul dans le cas $n = 3$ donne la même réciproque: on trouve un élément $v = x_1^2x_2 + a_1x_1^2 + (a_1^2 + a_2)x_1 + a_2x_2 \neq 0$ tel que $\mathrm{Fix}(\mathrm{S}_3) = \mathbf{A} \oplus \mathrm{Ann}_{\mathbf{A}}(\langle 2, \mathrm{di}\, f\rangle)\, v$. Par contre pour $n \geq 4$, la situation se complique.

5 Séparabilité de $\mathrm{Adu}_{\mathbf{A},f}$

Dans cette section on suppose que $\mathrm{disc}\, f$ est un élément **inversible** de $\mathbf{A}$.

Le lemme suivant comme moyen de prouver constructivement le théorème 5.2 a été suggéré par Thierry Coquand.

Lemme 5.1. *Soit $\phi : \mathbf{A} \to \mathbf{C}$ une algèbre dans laquelle « f se factorise complètement », c'est-à-dire $\phi(f) = \prod_{i=1}^{n}(T - u_i)$. Pour tout $\sigma \in \mathrm{S}_n$ notons $\phi_\sigma : \mathbf{B} \to \mathbf{C}$ l'unique homomorphisme de $\mathbf{A}$-algèbres qui envoie chaque x_i sur $u_{\sigma i}$. Soit $y \in \mathbf{B}$ tel que $\phi_\sigma(y) = 0$ pour tout $\sigma \in \mathrm{S}_n$, alors les coordonnées de y sur la base naturelle $\mathcal{B}(f)$ décrite dans le lemme 1.3 sont dans* $\mathrm{Ker}\, \phi$.

Démonstration. Nous donnons la preuve pour $n = 4$ et laissons le soin au lecteur de rédiger une preuve formelle par récurrence .
On commence par remarquer que les $x_i - x_j$ sont inversibles pour $i \neq j$ et par suite les $u_i - u_j$ sont inversibles pour $i \neq j$. La base naturelle est formée par 24 éléments $x_1^{m_1}x_2^{m_2}x_3^{m_3}$ avec $0 \leq m_i \leq 4 - i$. Notons $y = a(x_1, x_2) + x_3c(x_1, x_2)$ avec a et c des polynômes formels

de degré ≤ 3 en X_1 et ≤ 2 en X_2. Notons $\overline{a}$ et $\overline{c}$ les images des polynômes formels a et c dans $\mathbf{C}$ par ϕ. Notre but est de montrer que $\overline{a}$ et $\overline{c}$ sont des polynômes identiquement nuls. En considérant pour σ d'une part l'identité et d'autre part la transposition qui échange 3 et 4, on obtient dans $\mathbf{C}$:

$$\overline{a}(u_1,u_2)+u_3\overline{c}(u_1,u_2)=0=\overline{a}(u_1,u_2)+u_4\overline{c}(u_1,u_2)$$

Puisque u_3-u_4 est inversible, on en déduit $\overline{a}(u_1,u_2)=0=\overline{c}(u_1,u_2)$.
La preuve qu'on vient de faire fonctionne aussi si on permute arbitrairement les x_i (on change alors de base naturelle), de sorte que $\overline{a}(u_i,u_j)=0=\overline{c}(u_i,u_j)$ chaque fois que $i\neq j$. On va montrer que $\overline{a}$ est identiquement nul, la même preuve s'appliquant à $\overline{c}$. On considère $\overline{a}(u_1,X_2)$: ce polynôme de degré ≤ 2 admet les trois racines u_2, u_3, u_4 et puisque les u_i-u_j sont inversibles la formule d'interpolation de Lagrange montre que $\overline{a}(u_1,X_2)$ est nul comme polynôme en X_2. Chacun de ses trois coefficients est un polynôme de degré ≤ 3 en X_1 qu'on évalue en u_1 (on obtient ainsi 12 coordonnées de y sur la base naturelle, les 12 autres correspondant au polynôme c). Notons $e(X_1)$ l'un de ces trois polynômes, lu dans $\mathbf{A}[X_1]$. La preuve que nous avons faite, montrant que $\overline{e}(u_1)=0$ fonctionne aussi si on permute arbitrairement les x_i. Donc $\overline{e}(u_i)=0$ pour tous les i. Encore une fois nous appliquons la formule d'interpolation de Lagrange et nous voyons que $\overline{e}(X_1)$ est identiquement nul. □

Théorème 5.2. *1. Le nilradical de* $\mathbf{B}$ *est l'idéal engendré par le nilradical de* $\mathbf{A}$*. En particulier, si* $\mathbf{A}$ *est réduite,* $\mathbf{B}$ *est réduite.*

2. Pour toute algèbre réduite $\mathbf{A}\xrightarrow{\rho}\mathbf{D}$*,* $\mathbf{B}\otimes_{\mathbf{A}}\mathbf{D}\simeq \mathrm{Adu}_{\mathbf{D},\rho(f)}$ *est réduite.*

Démonstration. Il suffit de montrer le point 1. Soit $\mathfrak{N}$ le nilradical de $\mathbf{B}$. Appliquons le lemme précédent avec $\mathbf{C}=\mathbf{B}/\mathfrak{N}$ et $y\in\mathbf{B}$ qui est nilpotent. L'élément y reste nilpotent si on le transforme par un élément de S_n. Le lemme s'applique: les coordonnées de y sur la base naturelle sont toutes dans $\mathfrak{N}\cap\mathbf{A}$. □

Le théorème suivant s'applique en particulier pour l'algèbre de décomposition universelle $\mathbf{B}$.

Théorème 5.3. (diagonalisation d'un quotient de Galois d'une algèbre de décomposition universelle)
Soit e *un idempotent galoisien de* $\mathbf{B}$*,* G *son stabilisateur et* $\mathbf{B}_1=\mathbf{B}/\langle 1-e\rangle$*. On note* $y_i=\pi(x_i)$ *la classe de* x_i *dans* $\mathbf{B}_1$*. Soit* $\phi:\mathbf{B}_1\to\mathbf{C}$ *un homomorphisme d'anneaux. On note* $u_i=\phi(y_i)$*. On considère* $\mathbf{C}_1=\mathbf{B}_1\otimes_{\mathbf{A}}\mathbf{C}$*. Pour tout* $\sigma\in G$ *notons* $\phi_\sigma:\mathbf{C}_1\to\mathbf{C}$ *l'unique homomorphisme de* $\mathbf{C}$*-algèbres qui envoie chaque* $y_i\otimes 1_{\mathbf{C}}$ *sur* $u_{\sigma i}$*. Soit* $\Phi:\mathbf{C}_1\to\mathbf{C}^{|G|}$ *l'homomorphisme de* $\mathbf{C}$*-algèbres défini par* $z\mapsto(\phi_\sigma(z))_{\sigma\in G}$*.*

1. Φ *est un isomorphisme:* $\mathbf{C}$ *diagonalise* $\mathbf{B}_1$*.*

2. En particulier $\mathbf{B}_1\otimes_{\mathbf{A}}\mathbf{B}_1$ *est isomorphe canoniquement à* $\mathbf{B}_1^{|G|}$*:* $\mathbf{B}_1$ *se diagonalise elle-même.*

Démonstration. Les deux algèbres sont des $\mathbf{C}$-modules projectifs de rang constant $|G|$ et Φ est une application $\mathbf{C}$-linéaire dont il suffit de démontrer la surjectivité. Dans $\mathbf{C}_1$ nous notons y_i à la place de $y_i \otimes 1_{\mathbf{C}}$ et u_i à la place de $1_{\mathbf{B}_1} \otimes u_i$. La surjectivité résulte par le théorème chinois de ce que les $\operatorname{Ker}\phi_\sigma$ sont deux à deux comaximaux: $\operatorname{Ker}\phi_\sigma$ contient $y_i - u_{\sigma i}$, $\operatorname{Ker}\phi_\tau$ contient $y_i - u_{\tau i}$, donc $\operatorname{Ker}\phi_\sigma + \operatorname{Ker}\phi_\tau$ contient les $u_{\sigma i} - u_{\tau i}$, et il y a au moins un indice i pour lequel $\sigma i \neq \tau i$ ce qui donne $u_{\sigma i} - u_{\tau i}$ inversible. □

6 Structure triangulaire des idéaux galoisiens

Nous démontrons dans cette section un résultat donné dans [1] et [8] en le généralisant un peu: notre théorème 6.1. Notre méthode de preuve est différente car elle ne s'appuie pas sur l'existence d'une cloture algébrique, et le cadre est plus général puisque nous avons à la base un anneau commutatif presque arbitraire à la place d'un corps.

Ce résultat affirme que la structure de l'idéal $\mathcal{J}(f)$, qui est une structure « triangulaire » (au sens de Lazard) lorsqu'on considère les modules de Cauchy comme générateurs, se retrouve pour tous les idéaux galoisiens de l'algèbre de décomposition universelle dans le cas d'un polynôme séparable.

Les anneaux que nous considérons sont les anneaux $\mathbf{A}$ qui vérifient la propriété suivante: l'anneau total des fractions de $\mathbf{A}$, $\operatorname{Frac}\mathbf{A}$, est zéro-dimensionnel. C'est notamment le cas des anneaux intègres, des anneaux zéro-dimensionnels et des anneaux nœthériens.

Nous aurons besoin des résultats suivants que nous utilisons librement dans la preuve du théorème.

- Si $(\mathbf{A}, \mathbf{C}, G)$ une algèbre galoisienne, $\mathbf{C}$ est un $\mathbf{A}$-module projectif de rang $|G|$, et $\mathbf{A}$ est facteur direct dans $\mathbf{C}$.
- Si $\mathbf{A}$ est zéro-dimensionnel, tout module projectif de rang constant est libre.
- Si $\mathbf{A}$ est zéro-dimensionnel, et si $N \subset \mathbf{A}^n$ est libre, il existe $M \subset \mathbf{A}^n$ libre tel que $\mathbf{A}^n = M \oplus N$ (théorème de la base incomplète).

Théorème 6.1. *Soit $(\mathbf{A}, \mathbf{C}, G)$ une algèbre galoisienne avec*

- $\mathbf{C} = \mathbf{A}[y_1, \ldots, y_n]$,
- *G opère sur $\{y_1, \ldots, y_n\}$ et*
- *les $y_i - y_j$ inversibles pour $i \neq j$.*

On suppose que l'anneau total des fractions de $\mathbf{A}$, $\operatorname{Frac}\mathbf{A}$, est zéro-dimensionnel. On note $G = G_0$, $G_i = \{\sigma \in G\,;\, \sigma(y_k) = y_k, \forall k \leq i\}$, $(i = 1 \ldots, n)$, et

$$r_i(T) = \prod_{\sigma \in G_{i-1}/G_i} (T - \sigma(y_i))$$

où G_{i-1}/G_i désigne un système de représentants des classes à gauche. Alors:

- $\mathbf{A}[y_1,\dots,y_i] = \mathrm{Fix}(G_i)$ *et* $G_i = \mathrm{Stp}(\mathbf{A}[y_1,\dots,y_i])$.
- $r_i(T)$ *est un polynôme unitaire de degré* $(G_{i-1} : G_i)$ *à coefficients dans* $\mathbf{A}[(y_k)_{k<i}]$, *on note* $R_i(X_1,\dots,X_i)$ *un polynôme unitaire de degré* $(G_{i-1} : G_i)$ *de* $\mathbf{A}[X_1,\dots,X_i]$ *tel que* $R_i(y_1,\dots,y_{i-1},X_i) = r_i(X_i)$.
- *L'idéal* $\mathfrak{a}_i = \mathfrak{a} \cap \mathbf{A}[X_1,\dots,X_i]$ *est engendré par* $R_1(X_1),\dots,R_i(X_1,\dots,X_i)$.

En conséquence chacune des algèbres $\mathbf{A}[y_1,\dots,y_i]$ *est à la fois un* $\mathbf{A}[y_1,\dots,y_{i-1}]$*-module libre de rang* $(G_{i-1} : G_i)$ *et un* $\mathbf{A}$*-module libre de rang* $(G : G_i)$, *et chacun des idéaux* $\mathfrak{a}_i$ *est un idéal triangulaire (au sens de Lazard) de* $\mathbf{A}[X_1,\dots,X_i]$.

Démonstration. Le groupe G_1 est un groupe séparant d'automorphismes de l'anneau $\mathbf{C}$. On note $\mathbf{A}_1$ l'anneau des points fixes de G_1. On sait que $\mathbf{C}$ est un $\mathbf{A}_1$-module projectif de rang constant $|G_1|$ et que $\mathbf{A}[y_1] \subset \mathbf{A}_1$. En outre $\mathbf{A}_1$ est facteur direct dans $\mathbf{C}$, donc est un $\mathbf{A}$-module projectif de rang constant $|G|/|G_1|$. Les coefficients de $r_1(T)$ sont fixes par G, donc dans $\mathbf{A}$, parce que les $\sigma(y_1)$ pour $\sigma \in G/G_1$ parcourent l'orbite de y_1 sous G. En outre $\deg r_1 = (G : G_1)$, de sorte que $\mathbf{A}[X_1]/\langle r_1(X_1)\rangle$ est libre de rang $(G : G_1)$. L'idéal $\mathfrak{a}_1$ est formé par tous les $R \in \mathbf{A}[X_1]$ qui annulent y_1. Un tel polynôme R vérifie $R(\sigma(y_1)) = 0$ pour tout $\sigma \in G/G_1$ parce que ses coefficients sont dans $\mathbf{A}$ et que σ fixe tous les éléments de $\mathbf{A}$. Donc R est multiple des $(T - \sigma(y_1))$. Or les idéaux $\langle T - y_i\rangle$ sont deux à deux comaximaux (parce que les $y_i - y_j$ sont inversibles), et l'intersection d'idéaux deux à deux comaximaux est égale à leur produit, donc R est multiple de r_1. Ainsi $\mathfrak{a}_1 = \langle r_1(X_1)\rangle$ et $\mathbf{A}[y_1]$ est libre de rang $(G : G_1)$.
On a donc la situation suivante:

- $\mathbf{A}_1$ est un $\mathbf{A}$-module projectif de rang constant $|G|/|G_1|$,
- $\mathbf{A}[y_1]$ est libre de rang $|G|/|G_1|$,
- $\mathbf{A}[y_1] \subset \mathbf{A}_1$.

Supposons maintenant l'anneau $\mathbf{A}$ zéro-dimensionnel. Alors $\mathbf{A}_1$ est libre sur $\mathbf{A}$, et $\mathbf{A}[y_1] = \mathbf{A}_1$ par le théorème de la base incomplète. Donc $\mathbf{A}[y_1] = \mathbf{A}_1 = \mathrm{Fix}(G_1)$ et $(\mathbf{A}[y_1], \mathbf{C}, G_1)$ est une algèbre galoisienne. Alors $\mathbf{C} = \mathbf{A}_1[y_2,\dots,y_n]$ avec G_1 qui opère sur $\{y_2,\dots,y_n\}$ et les $y_i - y_j$ inversibles. Tout le raisonnement précédent fonctionne à l'identique en remplaçant $\mathbf{A}$ par $\mathbf{A}_1$, G par G_1, y_1 par y_2 et G_1 par G_2. On termine donc par récurrence.

Passons au cas général. Nous avons de nouveau $\mathbf{A}[y_1] \simeq \mathbf{A}[X_1]/\langle r_1(X_1)\rangle$ et $\mathbf{A}[y_1] \subset \mathbf{A}_1 = \mathrm{Fix}(G_1)$. Notons S l'ensemble des éléments réguliers de $\mathbf{A}$ et $\mathbf{F} = \mathrm{Frac}\,\mathbf{A} = S^{-1}\mathbf{A}$. Remarquons que puisqu'un élément régulier de $\mathbf{A}$ est régulier dans $\mathbf{C}$ on a $\mathbf{C} \subset S^{-1}\mathbf{C} = \mathbf{C} \otimes_{\mathbf{A}} \mathbf{F}$. Le cas zéro-dimensionnel implique que $S^{-1}\mathbf{A}_1 = S^{-1}\mathbf{A}[y_1]$, et notre objectif est de montrer l'égalité $\mathbf{A}_1 = \mathbf{A}[y_1]$. Soit donc $z \in \mathbf{A}_1$ et $s \in S$ tel que $sz \in \mathbf{A}[y_1]$: $sz = c_0 + c_1 y_1 + \cdots + c_{d_1-1} y_1^{d_1-1}$. Posons $s_k = \sum_{\sigma \in G/G_1} \sigma(y_1)^k$. Ce sont les sommes de Newton pour le polynôme $r_1 = R_1$, donc des éléments de $\mathbf{A}$. On a

$$sz y_1^k = c_0 y_1^k + c_1 y_1^{k+1} + \cdots + c_{d_1-1} y_1^{k+d_1-1}.$$

donc pour un $\sigma \in G/G_1$, si $\sigma(y_1) = y_\ell$:

$$s\sigma(zy_1^k) = c_0 y_\ell^k + c_1 y_\ell^{k+1} + \cdots + c_{d_1-1} y_\ell^{k+d_1-1}.$$

Comme $zy_1^k \in \mathbf{A}_1$ on a $\sum_{\sigma\in G/G_1} \sigma(zy_1^k)$ fixe par G donc dans $\mathbf{A}$ et $s\sum_{\sigma\in G/G_1} \sigma(zy_1)^k = c_0 s_k + \cdots + c_{d_1-1} s_{k+d_1-1} \in s\mathbf{A}$. Sous forme matricielle:

$$\begin{bmatrix} s_0 & s_1 & s_2 & \cdots & s_{d_1-1} \\ s_1 & s_2 & & \cdots & s_{d_1} \\ \vdots & & & & \vdots \\ s_{d_1-1} & \cdots & \cdots & \cdots & s_{2d_1-2} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{d_1-1} \end{bmatrix} \in s\mathbf{A}^{d_1\times 1}$$

or le déterminant de la matrice carrée au premier membre est égal au discriminant de r_1 donc est inversible. Ainsi les c_j sont tous multiples de s.

Nous terminons en vérifiant que $\mathbf{A}[y_1]$ vérifie bien l'hypothèse du théorème, ce qui permet de faire fonctionner la récurrence. En effet, puisque $\mathbf{A}[y_1]$ est libre sur $\mathbf{A}$ les éléments réguliers de $\mathbf{A}$ sont réguliers dans $\mathbf{A}[y_1]$ et l'anneau total des fractions de $\mathbf{A}[y_1]$ contient $\mathbf{F}[y_1] \simeq \mathbf{F}[X_1]/\langle r_1(X_1)\rangle$, lequel est zéro-dimensionnel, donc égal à son anneau total des fractions. □

Le théorème 6.1 s'applique pour l'algèbre de décomposition universelle dans la situation suivante:
On suppose le polynôme f séparable. On considère un idempotent galoisien $e = 1 - s$ et l'idéal galoisien correspondant $\mathfrak{b} = \langle s\rangle$. On pose

$$\mathbf{C} = \mathbf{B}/\mathfrak{b} = \mathbf{A}[X_1, \ldots, X_n]/\mathfrak{a} = \mathbf{A}[y_1, \ldots, y_n]$$

avec:

- y_i est la classe de x_i modulo $\mathfrak{b}$ ou de X_i modulo $\mathfrak{a}$,
- $\mathfrak{a} = \mathcal{J}(f) + \langle S\rangle$ si $S \in \mathbf{A}[X_1, \ldots, X_n]$ et $s = S(x_1, \ldots, x_n)$.

On note $G = G_0 = \mathrm{St}(e) = \mathrm{St}(\mathfrak{b}) \subset \mathrm{S}_n$, on le considère comme un groupe de $\mathbf{A}$-automorphismes de $\mathbf{C}$.

On sait alors que $(\mathbf{A}, \mathbf{C}, G)$ est une algèbre galoisienne. En effet $\mathbf{A} = \mathrm{Fix}(G)$ et un élément σ de G distinct de l'identité ne fixe pas tous les y_i et donc l'un des $y_i - \sigma(y_i)$ engendre l'idéal $\langle 1\rangle$ de $\mathbf{C}$ car les $y_i - y_j$ sont inversibles pour $i \neq j$.

7 Corps des racines

Dans cette section, nous remplaçons l'anneau $\mathbf{A}$ par un corps discret $\mathbf{K}$ et nous expliquons comment l'algèbre de décomposition universelle permet d'obtenir le corps des racines d'un polynôme, ou au moins un substitut constructif de ce dernier.

Une $\mathbf{K}$-algèbre est dite *finie* si c'est un $\mathbf{K}$-espace vectoriel de type fini (en mathématiques constructives cela n'implique pas qu'on connaisse une base de l'espace vectoriel), *strictement finie* si c'est un $\mathbf{K}$-espace vectoriel de dimension finie.

Rappelons que les quotients de l'algèbre de décomposition universelle $\mathbf{B} = \mathrm{Adu}_{\mathbf{K},f}$ sont des $\mathbf{K}$-algèbres finies et toute $\mathbf{K}$-algèbre finie est un annneau *zéro-dimensionnel*.

7.1 f arbitraire

En mathématiques classiques un corps des racines pour un polynôme unitaire f sur un corps discret $\mathbf{K}$ est obtenu en quotientant l'algèbre de décomposition universelle $\mathrm{Adu}_{\mathbf{K},f}$ par un idéal $\sqrt{\langle 1-e\rangle}$ où e est un idempotent indécomposable (qui existe d'après le théorème 3.4, ou bien simplement en considérant un idéal non nul dont la dimension comme $\mathbf{K}$-espace vectoriel est minimale).

En mathématiques constructives on ne dispose pas toujours d'un tel idempotent. Le théorème suivant explique comment contourner la difficulté que pose la non existence du corps des racines en mathématiques constructives.

Théorème 7.1. *Soit $(z_i)_{i\in I}$ une famille finie d'éléments de $\mathbf{B} = \mathrm{Adu}_{\mathbf{K},f}$. Il existe un idempotent galoisien e_1 de $\mathbf{B}$ tel que chaque $\pi(z_i)$ est nul ou inversible dans l'algèbre quotient $\mathbf{B}_1 = \mathbf{B}\Big/\sqrt{\langle 1-e_1\rangle}$ (π est la projection canonique $\mathbf{B}\to\mathbf{B}_1$).*

Démonstration. Puisque $\mathbf{B}$ est zéro-dimensionnel on peut pour chaque $i\in I$ calculer un idempotent $g_i\in\mathbf{B}$ tel que z_i est inversible modulo $1-g_i$ et nilpotent modulo g_i. Appliqué à la famille des g_i le théorème 3.4 donne un idempotent galoisien e_1, tel que pour chaque i, $1-e_1$ divise g_i ou $1-g_i$. Donc dans l'algèbre quotient $\mathbf{B}_1 = \mathbf{B}/\langle 1-e_1\rangle$ chaque $\pi(z_i)$ est nilpotent ou inversible. □

Le théorème d'unicité du corps des racines admet la version constructive suivante, qui découle du théorème 3.4:

Théorème 7.2. *Soient deux $\mathbf{K}$-algèbres strictement finies $\mathbf{A}_1$ et $\mathbf{A}_2$ non nulles pour lesquelles f se décompose en produit de facteurs linéaires dans $\mathbf{C}_1 = \mathbf{A}_1/\sqrt{0}$ et $\mathbf{C}_2 = \mathbf{A}_2/\sqrt{0}$. On suppose en outre que $\mathbf{C}_1$ et $\mathbf{C}_2$ sont engendrées par les zéros correspondants de f. Alors il existe une $\mathbf{K}$-algèbre $\mathbf{C} = \mathbf{B}\Big/\sqrt{\langle 1-e\rangle}$ (e idempotent galoisien) avec les mêmes propriétés, et deux entiers r_i tels que $\mathbf{C}_1\simeq\mathbf{C}^{r_1}$ et $\mathbf{C}_2\simeq\mathbf{C}^{r_2}$.*

7.2 f séparable

Ici nous donnons une preuve constructive d'un résultat classique.

Notez que le rsultat fondamental suivant est obtenu sans utiliser le corps des racines (une approximation convenable de ce corps suffit).

Théorème 7.3. *Soit $\mathbf{K}$ un corps discret et $f\in\mathbf{K}[X]$ un polynôme séparable. Alors l'algèbre de décomposition universelle $\mathbf{B} = \mathrm{Adu}_{\mathbf{K},f}$ est séparable (i.e., tout élément annule un polynôme séparable de $\mathbf{K}[T]$).*

Démonstration. Si $\mathbf{K}$ est de caractéristique nulle, un polynôme est séparable si et seulement si il est sans facteur carré. Le fait que $\mathbf{B}$ est réduite implique alors que le polynôme minimal de tout élément de $\mathbf{B}$ est séparable.
Dans le cas général, la preuve est un peu plus compliquée. Soit $z\in\mathbf{B}$. Appliqué à la famille des $\sigma(z)-\tau(z)$ ($\sigma\neq\tau$ dans S_n) le théorème 7.1 donne un idempotent galoisien e_1 tel que dans l'algèbre quotient $\mathbf{B}_1 = \mathbf{B}/\langle 1-e_1\rangle$ chaque $\pi_1(\sigma(z)-\tau(z))$ est nul ou inversible (π_1

est la projection canonique $\mathbf{B} \to \mathbf{B}_1$). On rappelle que d'après les théorèmes 2.3 et 4.3, le stabilisateur G de e_1 opère sur $\mathbf{B}_1$ et les points fixes pour cette action sont exactement les éléments de $\mathbf{K}$ (identifié à $\pi_1(\mathbf{K})$). Soit alors $\{z_1, \ldots, z_t\}$ un ensemble de S_n-conjugués de z tels que $\{\pi_1(z_1), \ldots, \pi_1(z_t)\}$ soit l'orbite de $\pi_1(z)$ pour l'action de G. On considère le polynôme $P_1(T) = \prod_{i=1}^{t}(T - \pi_1(z_i))$. Ses coefficients sont fixés par G donc $P_1 \in \mathbf{K}[T]$. Et c'est un polynôme séparable par construction (comme polynôme dans $\mathbf{B}_1[T]$ son discriminant est inversible). Ainsi $\pi_1(z)$ annule le polynôme séparable $P_1(T) \in \mathbf{K}[T]$, c'est-à-dire encore $P_1(z) \in \langle 1 - e_1 \rangle$. Si $\{e_1, \ldots, e_k\}$ est l'orbite de e_1 sous S_n, on aura pour $i = 1, \ldots, k$ un polynôme séparable $P_i \in \mathbf{K}[T]$ avec $P_i(z) \in \langle 1 - e_i \rangle$. Finalement le ppcm P des P_i est lui-même un polynôme séparable de $\mathbf{K}[T]$ et $P(z) \in \bigcap_i \langle 1 - e_i \rangle = \langle 0 \rangle$. □

Algorithme 7.4. **Calcul d'un idéal galoisien et de son stabilisateur.**
Entrée : $(\mathbf{C}, G)$: quotient de Galois de $(\mathbf{B}, \mathrm{S}_n)$, y: élément ni nul ni inversible de $\mathbf{C}$; $S = \mathrm{St}(y)$.
Sortie : $\mathfrak{c}$: idéal galoisien contenant y, et tel que tout conjugué de y sous G est nul ou inversible dans $\mathbf{C}/\mathfrak{c}$; H: le sous groupe stabilisateur de $\mathfrak{c}$.
Variables locales : $\mathfrak{a}$: idéal de C; σ: dans G; L: liste d'éléments de G.
Début
  $\mathfrak{c} \leftarrow < y >$; $L \leftarrow [\,]$;
  pour σ **dans** G/S **faire**
G/S désigne un système de représentants des classes à gauche modulo S
    $\mathfrak{a} \leftarrow \mathfrak{c} + < \sigma(y) >$;
    si $\mathfrak{a} \neq 1$ **alors** $\mathfrak{c} \leftarrow \mathfrak{a}$; $L \leftarrow L \bullet [\sigma]$ **fin si** ;
  fin pour
  $H \leftarrow$ le sous-groupe de G formé par les α tels que: $\forall \sigma \in L$, $\alpha\sigma \in \bigcup_{\tau \in L} \tau S$.
Fin.

L'algèbre $\mathbf{B} = \mathrm{Adu}_{\mathbf{K},f}$ est réduite quand le polynôme f est séparable, donc tout idéal de type fini est engendré par un idempotent. Ce résultat implique qu'à partir d'un élément ni nul ni inversible y de $\mathbf{B}$ on peut calculer un idéal galoisien $\mathfrak{c}$ tel que $y \in \mathfrak{c}$ et tout conjugué de y sous S_n est nul ou inversible dans $\mathbf{B}/\mathfrak{c}$: $\mathfrak{c}$ est un idéal strict engendré par y et le plus grand nombre possible de conjugués de y.

En outre, le résultat reste le même si nous considérons une algèbre galoisienne $(\mathbf{K}, \mathbf{C}, G)$ qui est un quotient de Galois de $(\mathbf{K}, \mathbf{B}, \mathrm{S}_n)$.

Notons que l'algorithme 7.4 calcule un idéal galoisien $\mathfrak{c}$ engendré par un idempotent galoisien e_1 qui serait construit par l'algorithme 3.5 à partir d'un idempotent e tel que $(1-e)\mathbf{C} = y\mathbf{C}$. Ainsi $(\mathbf{C}/\mathfrak{c}, H)$ est une nouvelle approximation du corps de racines de f et de son groupe de Galois, meilleure que la précédente approximation $(\mathbf{C}, G)$.

Bibliographie

[1] Aubry P., Valibouze A. *Using Galois Ideals for Computing Relative Resolvents.* J.

Symbolic Computation, **30**, 635–651, (2000).

[2] BISHOP E., BRIDGES D. *Constructive Analysis*. Springer-Verlag (1985).

[3] BOURBAKI *Algèbre. Chap 4 à 7*. Masson. Paris (1981).

[4] DELLA DORA J., DICRESCENZO C., DUVAL D. *About a new method for computing in algebraic number fields*. In Caviness B.F. (Ed.) EUROCAL '85. Lecture Notes in Computer Science 204, 289–290. Springer (1985).

[5] DEMEYER F., INGRAHAM E. *Separable algebras over commutative rings*. Springer Lecture Notes in Mathematics 181 (1971).

[6] DÍAZ TOCA G. *Galois Theory, Splitting fields and Computer Algebra*. à paraître Journal of Symbolic Computation (2005).

[7] DUCOS L. *Thèse doctorale*. Poitiers (2000).

[8] DUCOS L. *Construction de corps de décomposition grâce aux facteurs de résolvantes. (French) [Construction of splitting fields in favour of resolvent factors]*. Communications in Algebra **28** no. 2, 903–924 (2000).

[9] EKEDAHL E., LASKOV D. *Splitting algebras, symmetric functions ans Galois Theory.* Journal of Algebra and its Applications, **4** (1), 59–76, (2005).

[10] MINES R., RICHMAN F., RUITENBURG W. *A Course in Constructive Algebra.* Universitext. Springer-Verlag, (1988).

Gema M Díaz-Toca
Dpto. de Matemática Aplicada, Universidad de Murcia, Espagne
gemadiaz@um.es
Henri Lombardi
Laboratoire de Mathématiques de Besançon, Université de Franche-Comté, France
henri.lombardi@univ-fcomte.fr
Claude Quitté
Laboratoire de Mathématiques SP2MI, Université de Poitiers, France
claude.quitte@math.univ-poitiers.fr

An introspective algorithm for the integer determinant

Jean-Guillaume Dumas Anna Urbańska

Abstract

We present an algorithm computing the determinant of an integer matrix A. The algorithm is *introspective* in the sense that it uses several distinct algorithms that run in a concurrent manner. During the course of the algorithm partial results coming from distinct methods can be combined. Then, depending on the current running time of each method, the algorithm can emphasize a particular variant. With the use of very fast modular routines for linear algebra, our implementation is an order of magnitude faster than other existing implementations. Moreover, we prove that the expected complexity of our algorithm is only $O\left(n^3(\log(n)+\log(\|A\|))^2\log(n)\right)$ *bit operations*, where $\|A\|$ is the largest entry in absolute value of the matrix.

1 Introduction

One has many alternatives to compute the determinant of an integer matrix. Over a field, the computation of the determinant is tied to that of matrix multiplication via block recursive matrix factorizations [12]. On the one hand, over the integers, a naïve approach would induce a coefficient growth that would render the algorithm not even polynomial. On the other hand, over finite fields, one can nowadays reach the speed of numerical routines [9]. The classical approach is thus to reduce the computation modulo some primes of constant size and to recover the integer determinant from the modular computations. For this, at least two variants are possible: Chinese remaindering and p-adic lifting. The first variant requires either a good *a priori* bound on the size of the determinant or an early termination probabilistic argument [10, §4.2]. It thus achieves an *output dependant* bit complexity of $O\left(n^\omega \log(|det(A)|)\right)$ where ω is the exponent of matrix multiplication (3 for the classical algorithm, and 2.375477 for the Coppersmith-Winograd method). Of course, with the coefficient growth, the determinant size can be as large as $\Omega(n\log(n))$ (Hadamard's bound) thus giving a large worst case complexity.

Now the second variant uses system solving and p-adic lifting [4] to get an approximation of this determinant with a $O\left(n^3(\log(n)+\log(\|A\|))^2\right)$ bit complexity [16]. Indeed, every integer matrix is unimodularly equivalent to a diagonal matrix $S = diag\{s_1,\ldots,s_n\}$ with $s_i|s_{i+1}$. This means that there exist integer matrices U, V with $\det U, \det V = \pm 1$, such that $A = USV$. The s_i are called the invariant factors of A. Then, solving a system with a random right hand side will reveal s_n as the common denominator of the solution vector entries with high probability.

The idea of [1] is thus to combine both approaches, i.e. to approximate the determinant by p-adic lifting and recover only the remaining part $(\det(A)/s_n)$ via Chinese remaindering. Then G. Villard remarked that at most $O(\sqrt{n})$ invariant factors can be distinct and that, in general, only the last $O(\log(n))$ of those are nontrivial [11]. This remark, together with a preconditioned p-adic solving computing the i-th invariant factor enable them to produce a $O^{\sim}(n^{2+\omega/2})$ worst case algorithm, where $O^{\sim}$ hides some logarithmic factors, and an algorithm with an expected $O(n^3(\log(n) + \log(\|A\|))^2 \log^2(n))$ complexity. Note that the actual best worst case complexity algorithm is $O^{\sim}(n^{2.697263} \log(\|A\|))$, which is $O^{\sim}(n^{3.2} \log(\|A\|))$ without fast matrix multiplication, by [14]. Unfortunately, these last two worst case complexity algorithms, though asymptotically better, are not the fastest for the generic case or for the actual matrix sizes. The best expected complexity algorithm is a Las Vegas algorithm of Storjohann [18] which uses an expected number of $O^{\sim}(n^{\omega} \log \|A\|)$ bit operations. In section 5 we compare the performance of this algorithm to ours, based on experimental results of [19].

In this paper, we propose a new way to extend the idea of [17, 20] to get the last consecutive invariant factors with high probability in section 3.2. Then we combine this with the scheme of [1]. This combination, is made in an adaptive way. This means that the algorithm will choose the adequate variant at run-time, depending on discovered properties of its input. More precisely, in section 4, we propose an algorithm which uses timings of its first part to choose the best termination. This particular kind of adaptation was somewhat introduced in [15] as introspective ; we use here the more specific definition of [3]. This enables us to prove in section 4.1 an expected complexity of $O\left(n^3(\log(n) + \log(\|A\|))^2 \log(n)\right)$ bit operations, gaining a $\log(n)$ factor and improving the constants from [11]. Moreover, we are able to detect the worst cases during the course of the algorithm thus enabling us to switch to the asymptotically fastest method. In general this last switch is not required and we show in section 5 that when used with the very fast modular routines of [7, 9] and the LinBox library [8], our algorithm can be an order of magnitude faster than other existing implementations.

2 Base Algorithms and Procedures

In this section we present the procedures in more detail and describe their probabilistic behavior. We start by a brief description of the properties of the Chinese Remaindering loop (CRA) with early termination (ET) (see [5]), then proceed with the *LargestInvariantFactor* algorithm to compute s_n (see [1, 11, 17]). We end the section with a summary of ideas of Abbott *et al.* [1], Eberly *et al.* and Saunders *et al.* [17].

2.1 Output dependant Chinese Remaindering Loop (CRA)

CRA is a procedure based on the Chinese remainder theorem. Determinants are computed modulo several primes p_i. Then the determinant is reconstructed modulo $p_0 \cdots p_{n-1}$ in the symmetric range via the Chinese reconstruction. The integer value of the determinant is thus computed as soon as the product of the p_i exceeds $2|\det(A)|$. We know that the product is big enough if it exceeds some upper bound on this value or, probabilistically,

if the reconstructed value remains identical for several successive additions of modular determinants. The principle of early termination (ET) is thus to stop the reconstruction before reaching the upper bound, as soon as the determinant remains the same for several steps [5].
Algorithm 2.1 is an outline of a procedure to compute the determinant using CRA loops with early termination, correctly with probability $1-\epsilon$. We start with a lemma.

Lemma 2.1. *Suppose that primes p_i greater than $l \geqslant 4$ are randomly sampled form a set P, and let r_n be the value of the determinant modulo $p_0 \cdots p_n$ computed in the symmetric range. Then*

(i) $r_n = det(A)$, *if* $n \geqslant N = \begin{cases} \lceil \log_l(|\det(A)|) \rceil & \text{if } \det(A) \neq 0 \\ 0 & \text{whenever } \det(A) = 0 \end{cases}$;

(ii) if $r_n \neq \det(A)$ *then there are at most* $R = \lceil \log_l(\frac{|\det(A) - r_n|}{p_0 \cdots p_n}) \rceil$ *primes* p_{n+1} *such that* $r_n = \det(A)$ *mod* $p_0 \cdots p_n p_{n+1}$;

(iii) if $r_n = r_{n+1} = \cdots = r_{n+k}$ *then* $Prob(r_n \neq \det(A)) < \epsilon$ *if only* $k \geqslant \lceil \log(1/\epsilon)/\log(\frac{P'}{\log_l(H)}) \rceil$, *where* $P' = |P| - \lceil \log(H)/\log(l) \rceil$ *and* H *is an upper bound for the determinant (e.g.* H *can be the Hadamard's bound:* $|det(A)| \leqslant (\sqrt{n}\|A\|)^n$*).*

Proof. For (i), notice that $-\lfloor \frac{p_0 \cdots p_n}{2} \rfloor \leqslant r_n < \lceil \frac{p_0 \cdots p_n}{2} \rceil$. Then $r_n = \det(A)$ as soon as $\lfloor \frac{p_0 \cdots p_n}{2} \rfloor \geqslant |\det(A)|$. With l being the lower bound for p_i this reduces to $n \geqslant \lceil \log_l |\det(A)| \rceil$ in the case when $\det(A) \neq 0$.
For (ii), we observe that $\det(A) = r_n + K p_0 \ldots p_n$ and it suffices to estimate the number of primes greater or equal l that divide K.
For (iii) we notice that k primes dividing K can be chosen with probability $\binom{R}{k} / \binom{|P| - n + 1}{k}$, which can be bounded by $(\frac{R}{P'})^k$ Since $R \leqslant \lceil \log_l(\frac{2H}{2}) \rceil$ we get the result. □
To compute the modular determinant in algorithm 2.1 we use the LU factorization and we refer to it as LU iteration. Early termination is particularly useful in the case when the computed determinant is much smaller than the *a priori* bound. The running time of this procedure is output dependant.

2.2 Largest Invariant Factor

A method to compute s_n for integer matrices was first stated by V. Pan [16] and later in the form of the *LargestInvariantFactor* procedure (LIF) in [1, 11, 5, 17]. The idea is to obtain a divisor of s_n by computing a rational solution of the linear systems $Ax = b$. If b is chosen at random from a sufficiently large set, then the computed divisor can be as close as possible to s_n with high probability. Indeed, with $A = USV$, we can equivalently solve $SVx = U^{-1}b$ for $y = Vx$, and then solve for x. As U and V are unimodular, the least common multiple of the denominators of x and y, d(x) and d(y) satisfies $d(x) = d(y)|s_n$. Thus, solving $Ax = b$ via p-adic lifting [4], enables us to get s_n with high probability at the cost of $O(n^3(\log(n) + \log(\|A\|))^2)$ independently of the size of s_n.

Algorithm 2.1 Early Terminated CRA

Require: An integer matrix A.
Require: $0 < \epsilon < 1$.
Require: A set P of random primes greater than l.
Ensure: The integer determinant of A, correct with probability at least $1 - \epsilon$.

1: $H = (\sqrt{n}\|A\|)^n$; $P' = |P| - \lceil \log(H)/\log(l) \rceil$; $i = 0$; // Hadamard's bound
2: **repeat**
3: Get a prime p_i from the set P; $P = P - \{p_i\}$
4: Compute $\det(A) \bmod p_i$; //via LU factorization of A modulo p_i.
5: Reconstruct r_i, the determinant modulo $p_0 \cdots p_i$; // by Chinese remaindering
6: $k = \max\{t : r_{i-t} = \cdots = r_i\}$; $R = \lceil \log_l \frac{H+|r_i|}{p_0 p_1 \ldots p_{i-k}} \rceil$
7: i++;
8: **until** $\frac{R(R-1)\ldots(R-k+1)}{(|P|-n)(|P|-n-1)\ldots(|P|-n-k+1)} < \epsilon$ or $\prod p_i > 2H + 1$

The algorithm takes as input parameters β and r which are used to control the probability of correctness. r is the number of successive solvings and β is the size of the set from which the values of a random vector b are chosen. With each system solving, the output $\tilde{s}_n$ of the algorithm is updated as the lcm of the current solution denominator $d(x)$ and the result obtained so far.
The following theorem characterizes the probabilistic behavior of the LIF procedure.

Theorem 2.2. *Let A be a $n \times n$ matrix, H its Hadamard's bound, r and β be defined as above. Then the output $\tilde{s}_n$ of Algorithm* LargestInvariantFactor *of [1] is characterized by the following properties.*

i) Let $r = 1$, p be a prime, $l \geqslant 1$, then $P(p^l | \frac{s_n(A)}{\tilde{s}_n}) \leqslant \frac{1}{\beta} \lceil \frac{\beta}{p^l} \rceil$;

ii) if $r = 2$, $\beta = \lceil (n+1)H \rceil$ then $\mathbf{E}\left(\log(\frac{s_n(A)}{\tilde{s}_n})\right) = O(1)$;

iii) if $r = 2$, $\beta = 6 + \lceil 2\log(\log(H)) \rceil$ then $s_n = \tilde{s}_n$ with probability at least 1/3;

iv) if $r = \lceil 2\log(\log(H)) \rceil$, $\beta \geqslant 2$ then $\mathbf{E}\left(\log(\frac{s_n(A)}{\tilde{s}_n})\right) = O(1)$;

v) if $r = \log(\log(H)) + \log(\frac{1}{\epsilon})$, $2 \mid \beta$ and $\beta \geqslant 2$ then $s_n(A) = \tilde{s}_n$ with probability at least $1 - \epsilon$;

Proof. The proofs of (i), (ii) and (iv) are in [1]. The proof of (iii) is in [11]. To prove (v) we slightly modify the proof of (iv) in the following manner. From (i) we notice that for every prime p dividing s_n, the probability that it divides the missed part of $s_n(A)$ satisfies:

$$P(p \mid \frac{s_n}{\tilde{s}_n}) \leqslant (\frac{1}{2})^r.$$

As there are at most $\log(H)$ such primes, we get

$$P(s_n = \tilde{s}_n) \geqslant 1 - \log(H)(1/2)^r \geqslant 1 - \log(H)2^{-\log(\log(H))-\log(\frac{1}{\epsilon})} = 1 - \log(H)\frac{1}{\log(H)}\epsilon.$$

□

2.3 Abbott-Bronstein-Mulders, Saunders-Wan and Eberly-Giesbrecht-Villard ideas

Now, the idea of [1] is to combine both the Chinese remainder and the LIF approach. Indeed, one can first compute s_n and then reconstruct only the remaining factors of the determinant by reconstructing $\det(A)/s_n$. The complexity of this algorithm is $O\left(n^3 \log(|\det(A)/s_n(A)|)\right)$ which is unfortunately $O^{\sim}(n^4)$ in the worst case. However, nothing is known about the algorithm expected complexity.
Now Saunders and Wan [17, 20] proposed a way to compute not only s_n but also s_{n-1} (which they call a bonus) in order to reduce the size of the remaining factors $d/(s_n s_{n-1})$. The complexity doesn't change.
Then, Eberly, Giesbrecht and Villard have shown that the expected number of non trivial invariant factors is small, namely less than $\lceil 3log_\lambda(n)\rceil + 29$ in general if the entries of the matrix are chosen in a set of λ consecutive integers [11]. As they also give a way to compute any $s_i(A)$ which leads to an algorithm with expected complexity $O(n^3(\log(n) + \log(\|A\|))^2 \log(n))\log_\lambda(n))$.
Our idea is to extend the method of Saunders and Wan to get the last $O(\log_\lambda n)$ invariant factors of A slightly faster than by [11]. Then, we are able to remove one of the $\log(n)$ factors of the expected complexity. Moreover, we will show in the following sections that this enables us to build an adaptive algorithm solving a minimal number of systems.
We should also mention, that it should be possible to change a $\log(n)$ factor in the expected complexity of [11] to a $\log\log(n)$ employing the bound for the expected number of invariant factors twice. Indeed their extra $\log(n)$ factor comes from the algorithm where n non trivial invariant factors are to be computed. But in the expected case, as they have only $\log(n)$ of those, this extra factor could be consequently reduced.

3 Computing the product of $O(\log(n))$ last invariant factors

3.1 On the number of invariant factors

The result in [11] says that a $n \times n$ matrix with entries chosen randomly and uniformly from a set of size λ has the expected number of invariant factors bounded by $\lceil 3\log(n)\rceil + 29$. In search for a sharpening of this result we prove the following theorems.

Theorem 3.1. *Let p be a prime. The expected number of non-trivial invariant factors divisible by p is at most 6.*

Theorem 3.2. *The expected number of nontrivial invariant factors is at most* $\log_\lambda(n) + \log_\lambda(\log_\lambda(n)+1) + 9$.

Both proofs can be found in the appendix A.

3.2 Extended Bonus Idea

In his thesis [20], Z. Wan introduces an idea of computing the penultimate invariant factor (i.e. s_{n-1}) of A while computing s_n using 2 system solvings. The additional cost is comparatively small, therefore s_{n-1} is referred to as a bonus. Here, we extend this idea to the computation of the $(n-k)$th factor with $(k+1)$ solvings.
Let $x^{(j)}$ be the rational solution to the equation $Ax^{(j)} = b^{(j)}$, where $b^{(j)}$ is a random vector. Then $x^{(j)}$ coordinates have a common denominator $\tilde{s}_n$ and we let $n^{(j)}$ denote the vector of numerators of $x^{(j)}$. Then $x^{(j)} = \frac{1}{\tilde{s}_n} n^{(j)}$ and $\gcd(n^{(j)}, \tilde{s}_n) = 1$.
Let B denote the $n \times (k+1)$ matrix $[b^{(j)}]_{j=1,\dots,k+1}$. Following Wan, we notice that $s_n(A)A^{-1}$ is an integer matrix, the Smith form of which is equal to

$$diag(\frac{s_n(A)}{s_n(A)}, \frac{s_n(A)}{s_{n-1}(A)}, \dots, \frac{s_n(A)}{s_1(A)}).$$

Therefore, we may compute $s_{n-k}(A)$ when knowing $s_{k+1}\left(s_n(A)A^{-1}\right)$. The trick is that the computation of A^{-1} is not required: we can perturb A^{-1} by right multiplying it by B. Then, $s_{k+1}(s_n(A)A^{-1}B)$ is a multiple of $s_{k+1}(s_n(A)A^{-1})$. Instead of $s_n(A)A^{-1}B$ we would prefer to use $\tilde{s}_n A^{-1}B$ which is already computed and equal to N, where $N = [n^{(j)}]_{j=1,\dots,k+1}$ is the matrix of numerators. The relation between A and N is as follows.

Lemma 3.3. *Let* $\tilde{s}_n^{-1}N$, $\gcd(\tilde{s}_n, N) = 1$ *be a solution to the equation* $AX = B$, *where* B *is* $n \times k$ *and the entries of* B *are uniformly and randomly chosen from the set* $\{0, 1, \dots \beta - 1\}$. *Then*

$$\frac{\tilde{s}_n}{\gcd(s_{i+1}(N), \tilde{s}_n)} | s_{n-i}(A), i = 1, 2 \dots, k.$$

Proof. The Smith forms of $s_n(A)A^{-1}B$ and N are connected by the relation $\frac{s_n(A)}{\tilde{s}_n} s_i(N) = s_i(s_n(A)A^{-1}B)$, $i = 1, \dots, (k+1)$. Therefore the quotient $\frac{s_n(A)}{s_{k+1}(s_n(A)A^{-1}B)}$ equals $\frac{\tilde{s}_n}{s_{k+1}(N)}$, and by taking $\frac{\tilde{s}_n}{\gcd(s_{k+1}(N), \tilde{s}_n)}$ one obtain an (integer) factor of $s_{n-k}(A)$. Moreover, the under-approximation is solely due to the choice of B.

□

Remark 3.4. Taking $\gcd(s_{k+1}(N), \tilde{s}_n)$ is necessary as $\frac{\tilde{s}_n}{s_{k+1}(N)}$ may be a rational number. Particularly, it happens when $s_1(N) = \gcd(N_{ij}) > 1$, and in this case, since $\gcd(\tilde{s}_n, N) = 1$, the impact of $s_1(N)$ on s_{n-k} is neglected. Moreover, this allows us to consider $p | \tilde{s}_n$ in all probability consideration throughout the paper.

In fact we are interested in computing the product $\pi_k = s_n s_{n-1} \cdots s_{n-k}(A)$ of the invariant factors of A. Then, following the idea of Abbott [1], we would like to reduce the computation of the determinant to the computation of $\frac{\det(A)}{\tilde{\pi}_k}$, where $\tilde{\pi}_k$ is a factor of π_k we have obtained.

We can compute $\tilde{\pi}_k$ as $\tilde{s}_n^{k+1}/\gcd(s_1 s_2 \cdots s_{k+1}(N), \tilde{s}_n^{k+1})$. The product of the $(k+1)$ first invariant factors of a matrix is equal to the gcd of all its $(k+1) \times (k+1)$ minors. In our approach, it suffices to compute $\lfloor n/(k+1) \rfloor$ of those. In the following lemmas we show that by repeating the choice of matrix B twice, we will omit only a finite number of bits in π_k. We start with a technical lemma, the proof of which is in Appendix B.

Lemma 3.5. *For $n \times n$ matrix A with entries chosen randomly and uniformly from the set $\{0, 1 \ldots S-1\}$, the probability that $p^l < S$ divides the determinant $\det(A)$ is at most $\frac{3}{p^l}$ for $p \neq 2, 3, 5$ and $\frac{4}{p^l}$ for $p = 2, 3, 5$ and $S \geqslant 81$.*

In the next lemma we discuss the impact of choosing only a few minors in $s_1 \ldots s_{k+1}(N)$ calculation. Here, $\mathrm{ord}_p(x)$ denotes the higher power of p dividing x.

Lemma 3.6. *Let N and B be as defined in lemma 3.3. Suppose that B is a random matrix with entries chosen uniformly form the set $\{0, 1, \ldots S-1\}, S \geqslant \max(H, 81)$ and $k = O(\log(n))$. Let $N = [N_1 | \ldots N_\mu | N']^T$ where N_i are $(k+1) \times (k+1)$ matrices, $\mu = \lfloor n/(k+1) \rfloor$. Then*

$$\sum_{l=1}^{\infty} \sum_{p^l | \tilde{s}_n} Prob\Big(\mathrm{ord}_p(\frac{\gcd_{i=1\ldots\mu}(\det(N_i))}{\gcd(minors(N))}) = l \Big) l \log(p) \ \in \ O(1). \tag{1}$$

Proof. The bound S on $\|B\|$ is chosen in the way that following remark 3.4 we can only consider $p \leqslant S$. Therefore from Lemma 3.5 we get that for $l \geqslant 1$

$$Prob(p^l | \det(N_i)) \leqslant \frac{\alpha}{p^l},$$

where $\alpha = 3$ for $p > 5$ and $\alpha = 4$ for $p = 2, 3, 5$.
Now the sum (1), which represents the size of the over-approximation due to partial gcd computation can be bounded by

$$\sum_{l=1}^{\infty} \sum_{p^l | \tilde{s}_n} Prob\Big(\mathrm{ord}_p(\frac{\gcd_{i=1\ldots\mu}(\det(N_i))}{\gcd(minors(N))}) = l \Big) l \log(p) \leqslant \sum_{l=1}^{\infty} \sum_{p^l | \tilde{s}_n} Prob\Big(p^l | \gcd_{i=1\ldots\mu}(\det(N_i)) \Big) \log(p)$$

$$\leqslant \log 2 (2 + \sum_{l=3}^{\infty} (\frac{4}{2^l})^\mu) + \log 3 (1 + \sum_{l=2}^{\infty} (\frac{4}{3^l})^\mu) + \log 5 (1 + \sum_{l=2}^{\infty} (\frac{4}{5^l})^\mu) + \sum_{p>5} \sum_{l=1}^{\infty} \log(p) (\frac{3}{p^l})^\mu$$

$$\leqslant 3 + 4 + 6 + 1 = 14$$

With μ being about $\frac{n}{O(\log(n))}$, the expected size of overestimation due to partial gcd calculation is $O(1)$. □

To consider the impact of the choice of B on our method we start with a remark, which is a small modification of [20, Lem. 5.17].

Remark 3.7. For every matrix M there exist a full rank $k \times n$, $k \leqslant n$, matrix V, such that $\mathrm{ord}_p(\frac{s_1 \cdots s_k(MB)}{s_1 \cdots s_k(M)})$ is less or equal $\mathrm{ord}_p(\det(VB))$.

Lemma 3.8. *Let A be an $n \times n$ integer matrix, $S \geqslant \max(H, 81)$ and B_i, $i = 1, 2$ be $n \times k$ matrices with the entries uniformly and randomly chosen from the set $\{0, 1, \dots S\}$. Then for $M = s_n(A)A^{-1}$*

$$\log\Big(\frac{\pi_k(A)}{s_n^k}\gcd(s_2\cdots s_{k+1}(MB_1), s_2\cdots s_{k+1}(MB_2))\Big) \in O(1).$$

Proof. From Remark 3.7 and Lemma 3.5, we have $Prob(B_i : \mathrm{ord}_p(\frac{s_1\cdots s_k(MB_i)}{s_1\cdots s_k(M)}) \leqslant l) \leqslant \frac{\alpha}{p^l}$, with $\alpha = 3$ for $p > 5$ and $\alpha = 4$ for $p = 2, 3, 5$. Now the expected size of the under-estimation is less than or equal to

$$\begin{aligned}\log(2)(1+\sum_{l=2}^{\infty}(\frac{4}{2^l})^2) + \log(3)(1+\sum_{l=2}^{\infty}(\frac{4}{3^l})^2) + \log(5)(1+\sum_{l=2}^{\infty}(\frac{4}{5^l})^2) + \sum_{5<p\leqslant H}\sum_{l=1}^{\infty}\log(p)(\frac{3}{p^l})^2 \\ \leqslant 3+4+6+\int_6^{\infty}\log(x)\frac{9}{x^2-1}dx,\end{aligned}$$

which is $O(1)$. □

It is worth noting that the above mentioned schemes could lead to an algorithm to compute several last (or first) invariant factors with possibly better probabilistic behavior and expected complexity than that of [11].

4 Introspective Algorithm

Now we should incorporate algorithm 2.1 and the ideas presented in sections 2.2 and 3.2 in the form of an introspective algorithm. Indeed, we give a recipe for an auto-adaptive program that implements several algorithms of diverse space and time complexities for solving a particular problem. The best path is chosen at run time, from a self-evaluation of the dynamic behavior (e.g. timings) while processing a given instance of the problem. This kind of auto-adaptation is called introspective in [3].

In the following, CRA loop refers here to algorithm 2.1, slightly modified to compute $\det(A)/K$. If we re-run the CRA loop, we use modular determinant results already computed to recover $\det(A)/K \bmod p$.

Theorem 4.1. *Algorithm 4.1 correctly computes the determinant with probability $1 - \epsilon$.*

Proof. Termination is possible only by the early terminated CRA loop or by the determinant algorithm used in the last step. The choice of k from theorem 2.1(iii) and the choice of the determinant algorithm from [13, 19] ensures that $1 - \epsilon$ probability is obtained. □

4.1 Complexity

The following theorem gives the complexity of the algorithm.

Algorithm 4.1 Extended Bonus Determinant Algorithm

Require: An integer $n \times n$ matrix A.
Require: $0 < \epsilon < 1$, an error tolerance.
Require: A stream S of random integers uniformly chosen from the set $\{0, 1 \ldots, \max(\lceil (n+1)H \rceil, 81)\}$, H - Hadamard's bound for A.
Require: A set P of random primes greater than l.
Ensure: The integer determinant of A, correct with probability at least $1 - \epsilon$.

$k = \log(1/\epsilon)/\lceil \log(\frac{P'}{\log_l(H)}) \rceil$; see Lem. 2.1(iii)
for $i = 1$ to k **do**
  run the CRA loop for $\det(A)$; //see Alg. 2.1
  if early terminated **then** Return determinant **end if**
end for
$i_{max} = \log_\lambda(n) + \log_\lambda(\log_\lambda(n) + 1) + 9)$; //see Theorem 3.2
for $j = 1, 2$ **do**
  $i = 0; \tilde{\pi}_{-1} = 1; K = 1;$
  while $i < i_{max}$ **do**
    Generate b_i a random vector of dimension n from the stream S;
    Compute $\tilde{s}_n$ by solving $Ax_i = b_i$; //see Section 2.2
    if $i = 0$ **then** $i = 1; \tilde{\pi}_0 = \tilde{s}_n;$
    else
      $N := \tilde{s}_n X$, where $X = [x_l]_{l=0,\ldots i}$; //see Section 3.2;
      $\tilde{\pi}_i = 0;$
      for $l = 1, \ldots \lfloor n/i \rfloor$ **do**
        $\tilde{\pi}_i = \gcd(\tilde{\pi}_i, \det(N_l))$, where N_l is the lth minor of N;
      end for
      i=i+1;
    end if
    $K = \mathrm{lcm}(\tilde{\pi}_{i-1}, K)$; $\tilde{\pi}_{i-1} = K;$
    Resume CRA looping on $d = \det(A)/K$; for at most the time of one system solving;
    if early terminated **then** Return $d \cdot K$; **end if**
    if $\tilde{\pi}_{i-1} = \tilde{\pi}_{i-2}$ **then**
      Resume CRA looping on $d = \det(A)/K$; for at most the time of $(i_{max} - i)$ system solvings;
      if early terminated **then** Return $d \cdot K$;
      else $i = i_{max}$; **end if**
    end if
  end while
end for
run an asymptotically better integer determinant algorithm;

Theorem 4.2. *The expected complexity of Algorithm 4.1 is*

$$\mathrm{O}^{\approx}\left(n^{\omega}\log(1/\epsilon)+n^3(\log n+\log(\|A\|))^2\log(n)\right)$$

where $\mathrm{O}^{\approx}$ *hides some* $\log(\log(n))$ *factors. The pessimistic complexity depends on the algorithm used in the last step.*

Proof. To analyze the complexity of the algorithm we would consider the complexity of each step. With k defined as in the algorithm, the complexity of initial CRA iteration is $O\left(n^{\omega}\log(1/\epsilon)\right)$. The system solving in the LIF algorithm is performed $2i_{max}$ times with $\log(\|B\|)=O(n\log(n))$, which results with a complexity of $O(i_{max}n^3(\log(n)+\log(\|A\|)^2)$. Considering the time limit, this is also the time of all CRA loop iterations. To compute $\tilde{\pi}_i$ by means of the CRA determinant algorithm, we need $O\Big(\lfloor n/i\rfloor i^4(\log(i)+n(\log(n)+\log(\|A\|)+\log(\|B\|))\Big)$ bit operations, which for $i=2,\ldots,i_{max}$ with i_{max} being $O(\log(n))$ is $\mathrm{O}^{\sim}(n^2)$ and thus negligible.
With the expected number of invariant factors bounded by $i_{\max}$ (see Thm.3.2), it is expected that the algorithm will return the result before the end of the *while* loop, provided that the under-estimation of $\tilde{\pi}_{i_{max}}$ is not too big. But by updating $\tilde{s}_n$ $O(\log(n))$ times and updating the product $\tilde{\pi}_{i_{max}}$ twice, it is expected that the overall under-estimation will be $O(1)$ (see Theorem 2.2(ii) and Lemma 3.8), thus it is possible to recover it by several CRA loop iterations.

□

5 Experiments and Further Adaptivity

The described algorithm is implemented in the LinBox exact linear algebra library [8]. In a preliminary version i_{max} is set to 2 or 1 and the switch in the last step is not implemented. This is however enough to evaluate the performance of the algorithm and to introduce further adaptive innovations.
Comparing the data from table 5 we notice that the algorithm with $i_{max}=1$ (which is in fact a slightly modified version of Abbott's algorithm [1]) runs better for small n. Those timings have been evaluated on a set of matrices which have the same Smith form as $diag\{1,2,..,n\}$ and the number of invariant factors of about $\frac{n}{2}$. For every matrix, with each step, the size of s_{n-i} decreases whilst the cost of its computation increases. This accounts for better performance of Abbott's algorithm, which computes only s_n, in the case of small n. For bigger n calculating s_{n-1} starts to pay out. The same situation repeats at each step.
The switch between winners can be explained by the fact that in some situations, obtaining s_{n-i} by LU-factorization (which costs $\frac{\log(s_{n-i})}{\log(l)}$ the time of LU) outperforms system solving. Then, this also holds for all consecutive factors and the algorithm basing on CRA wins. The condition can be checked *a posteriori* by approximating the time of LUs needed to compute the actual factor. We can therefore construct a condition that would allow us to turn to the CRA loop in the appropriate moment. This can be done by changing the condition in

n	$i_{max}=1$	$i_{max}=2$	n	$i_{max}=1$	$i_{max}=2$
100	0.17	0.22	300	5.65	5.53
120	0.29	0.33	350	9.76	9.64
140	0.48	0.55	400	14.99	14.50
160	0.73	0.78	600	57.21	54.96
180	1.07	1.16	800	154.74	147.53
200	1.49	1.51	1000	328.93	309.61
250	2.92	3.00	2000	3711.26	3442.29

Table 1: Comparison of the performance of Algorithm 4.1 with $i_{\max}$ set to 1 and 2 on engineered matrices.

line 24 ($\tilde{\pi}_{i-1} = \tilde{\pi}_{i-2}$) to

$$\log(\frac{\tilde{\pi}_{i-1}}{\tilde{\pi}_{i-2}}) \leqslant \frac{time(solving)}{time(LU)} \log(l),$$

if the primes used in the CRA loop are greater than l. This would result with a performance close to the best and yet flexible. If, to some extend, s_{n-i} could be approximated *a priori*, this condition could be checked before its calculation. This would require a partial factorization of s_{n-i+1} and probability considerations as in the appendix A and [11].
For a generic case of random dense matrices our observation is that the bound for the number of invariant factors is quite crude. Therefore the algorithm 4.1 is constructed in the way that minimizes the number of system solving to at most twice the actual number of invariant factors for a given matrix. Under the assumption that the approximations $\tilde{s}_n$ and $\tilde{\pi}_i$ are sufficient, this leads to a quick solution.
Indeed for random matrices, the algorithm nearly always stopped with early termination after one system solving. This together with fast underlying arithmetics of FFLAS [7] accounted for the superiority of our algorithm as seen in figure 5 where comparison of timings for different algorithms is presented.

6 Conclusions

In this paper we presented an algorithm computing the determinant of an integer matrix which expected time complexity is $O\left(n^3(\log(n)+\log(||A||))^2\log(n)\right)$. Our algorithm uses an introspective approach so that its actual running time is only $O\left(n^3(\log(n)+\log(||A||))^2 k\right)$ if the number k of invariant factors is smaller than *a priori* expected. Moreover, the adaptive approach allows us to switch to the algorithm with best worst case complexity if it happens that the number of nontrivial invariant factors is unexpectedly large. This adaptivity, together with very fast modular routines, allows us to produce an algorithm, to our knowledge, faster by at least an order of magnitude than other implementations.
Ways to further improve the running time are to reduce the number of iterations in the solvings or to group them in order to get some block iterations as is done e.g. in [2]. A

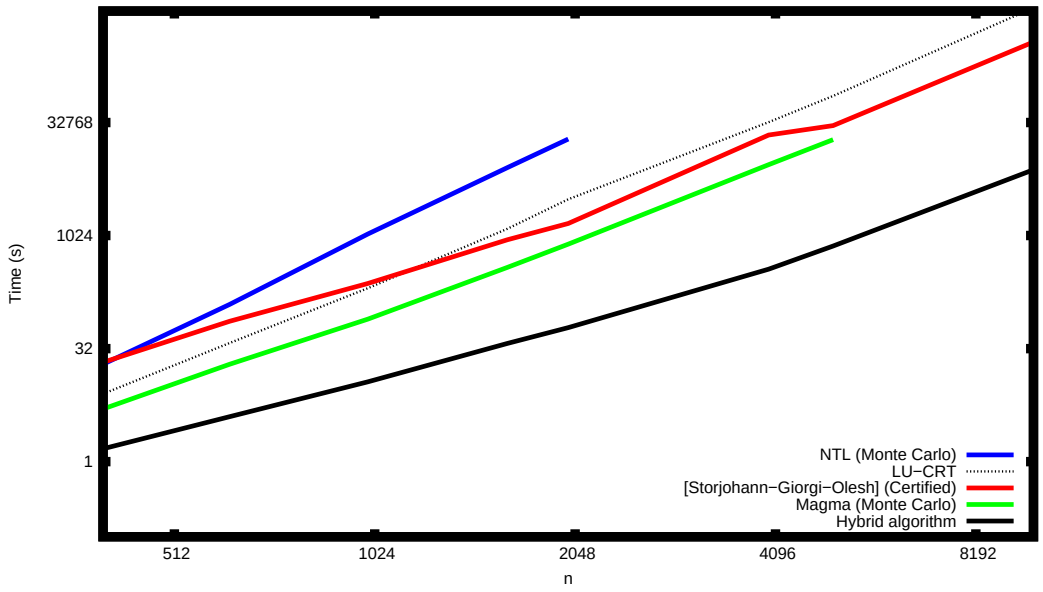


Figure 1: Comparison of our algorithm with other existing implementation. Tested on random dense matrices of the order 400 to 10000, with entries {-8,-7,. . . ,7,8} Using fast modular routines puts our algorithm several times ahead of the others. Scaling is logarithmic.

modification to be tested, is to try to reconstruct s_n with only some entries of the solution vector $x = \mathbf{n}/d$. Parallelization can also be considered to further modify the algorithm. Of course, all the LU iterations in one CRA step can be done in parallel. An equivalently efficient way is to perform several p-adic liftings in parallel, but with less iterations [6]. There the issue is to perform an optimally distributed early termination.

References

[1] J. Abbott, M. Bronstein, T. Mulders. Fast deterministic computation of determinants of dense matrices. In *Proc. of ACM International Symposium on Symbolic and Algebraic Computation (ISAAC'1999)*, 197-204, ACM Press, 1999.

[2] Z. Chen and A. Storjohann. A BLAS based C library for exact linear algebra on integer matrices. In *Proc. of ACM International Symposium on Symbolic and Algebraic Computation (ISAAC'2005)*, 92–99, ACM Press, 2005.

[3] V.-D. Cung, V. Danjean, J.-G. Dumas, T. Gautier, G. Huard, B. Raffin, C. Rapine, J.-L. Roch, D. Trystram, Adaptive and hybrid algorithms: classification and illustration on triangular system solving, in: *Proceedings of Transgressive Computing 2006, Granada, España.*

[4] J. Dixon. Exact Solution of Linear Equations Using P-Adic Expansions. In *Numer.Math.* 40(1), 137-141, 1982.

[5] J.G. Dumas, D. Saunders, G. Villard. On Efficient Sparse Integer Matrix Smith Normal Form Computations. In *Journal of Symbolic Computations.* 32 (1/2), 71-99, 2001.

[6] J.G. Dumas, W. Turner, Z. Wan. Exact Solution to Large Sparse Integer Linear Systems. *ECCAD'2002 : The 9th Annual East Coast Computer Algebra Day*, 2002.

[7] J.G. Dumas, T. Gautier, C. Pernet. FFLAS: Finite field linear algebra subroutines. *ISSAC'2002*. 2002.

[8] J.G. Dumas, T. Gautier, M. Giesbrecht, P. Giorgi, B. Hovinen, E. Kaltofen, D. Saunders, W. Turner, G. Villard. LinBox: A Generic Library for Exact Linear Algebra. *ICMS'2002 : International Congress of Mathematical Software* 2002.

[9] J.G. Dumas, P. Giorgi, C. Pernet. FFPACK: finite field linear algebra package. *ISSAC'2004*. 2004.

[10] J.G. Dumas, C. Pernet, Zhendong Wan. Efficient Computation of the Characteristic Polynomial. *ISSAC'2005*. 2005.

[11] W. Eberly, M. Giesbrecht, G. Villard. On computing the determinant and smith form of an integer matrix. In *Proc. 41st FOCS*, 675-687, 2000.

[12] O.H. Ibarra, S. Moran, R.Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3(1):45ŏ201356, Mar.1982.

[13] E. Kaltofen, G. Villard. Computing the sign or the value of the determinant of an integer matrix, a complexity survey. In *Journal of Computational and Applied Mathematics* 164(2004), 133-146 2004.

[14] E. Kaltofen, G. Villard. On the complexity of computing determinants. *Computational Complexity*, 31(3-4), pp 91–130, 2005.

[15] D. Musser. Introspective Sorting and Selection Algorithms. *Software—Practice and Experience*, 8(27), pp 983–993, 1997.

[16] V. Pan. Computing the determinant and the characteristic polynomial of a matrix via solving linear systems of equations. *Inform. Process. Lett.* 28(1988) 71-75. 1988.

[17] D. Saunders, Z. Wan. Smith Normal Form of Dense Integer Matrices, Fast Algorithms into Practice. *ISSAC 2004* 2004.

[18] A.Storjohann. The shifted number system for fast linear algebra on integer matrices. *Journal of Complexity*, 21(4), pp 609–650, 2005.

[19] A. Storjohann, P. Giorgi, Z. Olesh. Implementation of a Las Vegas integer Matrix Determinant Algorithm. *ECCAD'05: East Coast Computer Algebra Day*, 2005.

[20] Z. Wan. Computing the Smith Forms of Integer Matrices and Solving Related Problems. Ph.D. Thesis, U. of Delaware, USA, 2005.

A Proofs of theorems 3.1 and 3.2

In order to prove theorems stated in section 3.1, we will start with the following lemma.

Lemma A.1. *For $\lambda > 11$ the sum over primes p: $\sum_{8<p<\lambda} (\frac{1}{\lambda}\lceil\frac{\lambda}{p}\rceil)^j$ can be bounded by $(\frac{1}{2})^j$.*

Proof. We will consider primes from the interval $\frac{\lambda}{2^{k+1}} \leqslant p < \frac{\lambda}{2^k}$, $k = 0, 1, \ldots \max\{\lceil\log(\lambda)\rceil - 3, 2\}$ separately. For the kth interval $\lceil\frac{\lambda}{p}\rceil$ equals 2^{k+1}. In each interval there are at most $\lceil\frac{\lambda}{4}\rceil$ odd numbers and at most $\frac{\lambda}{4}$ primes. The reasoning goes as follows: if in the interval there are more than 3 odd numbers, at least one of them is divided by 3 and so does not count. For this to happen it is enough that $\lambda \geqslant 12$. We may therefore calculate:

$$\sum_{8<p<\lambda} (\frac{1}{\lambda}\lceil\frac{\lambda}{p}\rceil)^j \leqslant \sum_{k=0}^{\lceil\log(\lambda)\rceil-3} \frac{\lambda}{2^{k+2}}(\frac{2^{k+1}}{\lambda})^j \leqslant \frac{1}{2\lambda^{j-1}}(2^{\lceil\log(\lambda)\rceil-2})^{j-1} \leqslant (\frac{1}{2})^j.$$

□

Remark A.2. For $\lambda = 2^l$ we may consider primes $p > 4$.

Remark A.3. If we exclude {2,3,5,7,8,16}, we get the same bound for $\sum_{k\in\mathbf{N}} \sum_{8<p^k<\lambda} (\frac{1}{\lambda}\lceil\frac{\lambda}{p^k}\rceil)^j$.

Proof.[Theorem 3.1] The idea of the proof is similar to that of [11]. Let A be a random matrix with entries chosen uniformly and randomly from the set $\{0, 1, 2 \ldots \lambda - 1\}$. Let $MDep_i(p)$ denote an event that the submatrix A_i, including the first i columns of A mod p has rank at most $i - 2$ over $\mathbf{Z}_p$.
We are now going to find $P(MDep_i(p) \mid \neg MDep_{i-1}(p))$. Since the event $MDep_{i-1}(p)$ did not occur, A_{i-1} has p-rank $(i - 2)$ or $(i - 1)$. For $MDep_i$ it must be $(i - 2)$, thus, there exists a set of $(i - 2)$ rows R_{i-2} which has the full rank. Consider any row v_j that is left. If v_j is a combination of R_{i-2} the last (ith) entry of v_j is determined mod p. For $\lambda \geqslant p$ this means that the probability that v_j is a combination of R_{i-2} is at most λ^{-1}. For $p < \lambda$ this probability is $\frac{1}{\lambda}\lceil\frac{\lambda}{p}\rceil$ which is always less than or equal to $\frac{2}{p+1}$. As there are $n - i + 2$ vectors outside R_{i-2}, the probability that none of them is linearly independent with R_{i-2} over $\mathbf{Z}_p$ is at most $(\frac{2}{p+1})^{n-i+2}$ for $p < \lambda$ and $(\frac{1}{\lambda})^{n-i+2}$ for $p \geqslant \lambda$.
Since $P(MDep_i(p) \mid \neg MDep_{i-1}(p)) \geqslant P(MDep_i(p) \wedge \neg MDep_{i-1}(p))$, we have $P(MDep_i(p)) \leqslant P(\bigcup_{j=1}^{i}(MDep_j(p) \wedge \neg MDep_{j-1}(p)))$ which can be bounded by $(\frac{2}{p+1})^{n-i+2}\frac{p+1}{p-1}$ for $p < \lambda$ and $(\frac{1}{\lambda})^{n-i+2}\frac{\lambda}{\lambda-1}$ for $p \geqslant \lambda$.
Let the number of invariant factors divided by p be greater than j. The rank of A mod p is then at most $n - j$ over $\mathbf{Z}_p$. This in consequence means that for $j > 1$ the submatrix A_{n-j+2}

has the rank at most $n-j$, so the event $MDep_{n-j+2}(p)$ is fulfilled. Therefore matrix A has at least j invariant factors divided by p with probability at most

$$\begin{aligned}(\frac{2}{p+1})^j\frac{p+1}{p-1}, &\quad p<\lambda\\ (\frac{1}{\lambda})^j\frac{\lambda}{\lambda-1}, &\quad p\geqslant\lambda.\end{aligned}\tag{2}$$

Now the expected number of invariant factor divided by p is not greater than

$$\begin{aligned}3+3\sum_{j=3}^{j=n}(\frac{2}{3})^j=3+9(\frac{2}{3})^3\leqslant 6, &\quad p=2,\\ 1+\sum_{j=1}^{j=n}(\frac{2}{p+1})^j\frac{p+1}{p-1}=1+\frac{2(p+1)}{(p-1)^2}\leqslant 3, &\quad 2<p<\lambda,\\ 1+\frac{\lambda}{(\lambda-1)^2}<2, &\quad p\geqslant\lambda>2.\end{aligned}\tag{3}$$

□ *Proof.*[Theorem 3.2] In addition to $MDep_i(p)$ introduced earlier, let Dep_i denote an event that the first i columns of A are linearly independent and $MDep_i$, an event that either of $MDep_i(p)$ occurred. Recall that $P(Dep_1\vee MDep_1(p))\leqslant\lambda^{-n}$, and $P(Dep_i\mid\neg(Dep_{i-1}\vee MDep_{i-1}(p)))\leqslant\lambda^{-n+i-1}$.
To bound $P(MDep_i\mid\neg(Dep_{i-1}\vee MDep_{i-1}(p)))$ we sum the results for all primes. For $p<\lambda$, $i\leqslant n-1$ the sum can be bounded by

$$(\frac{2}{3})^{n-i+2}+\sum_{\lambda>p>8}(\frac{1}{\lambda}\lceil\frac{\lambda}{p}\rceil)^{n-i+2}\leqslant(\frac{2}{3})^{n-i+2}+(\frac{1}{2})^{n-i+2},$$

thanks to the lemma A.1.
For primes $p\geqslant\lambda$ we should estimate the number of primes dividing the $(i-1)$th minor. By the Hadamard's bound (notice that Dep_{i-1} does not hold), the minors are bounded in absolute value by $((i-1)\lambda^2)^{\frac{i-1}{2}}$. Therefore the number of primes $p\geqslant\lambda$ dividing the minor is at most $\frac{i-1}{2}(\log_\lambda(i-1)+2)$. Summarizing,

$$\begin{aligned}&P((MDep_i\wedge Dep_i)\mid\neg(Dep_{i-1}\vee MDep_{i-1}(p)))\\ &\qquad\leqslant(\frac{1}{\lambda})^{n-i+1}+(\frac{2}{3})^{n-i+2}+(\frac{1}{2})^{n-i+2}+\frac{i-1}{2}\,(\log_\lambda(i-1)+2)\,(\frac{1}{\lambda})^{n-i+2}\end{aligned}$$

for $2\leqslant i\leqslant n-1$.
By the same argument as in the previous proof

$$P(MDep_{n-j+2})\leqslant\lambda^{-n}+(\frac{1}{\lambda})^{j-1}\frac{\lambda}{\lambda-1}+3(\frac{2}{3})^j+2(\frac{1}{2})^j+\frac{n-j+1}{2}\,(\log_\lambda(n-i+1)+2)\,(\frac{1}{\lambda})^j\frac{\lambda}{\lambda-1}.\tag{4}$$

Similarly, the probability that the number of invariant factors at least j is greater than $P(MDep_{n-j+2})$.
To calculate the expected number of invariant factors we first consider the case

$$\frac{n-j+1}{2}\big(\log_\lambda(n-j+1)+2\big)(\frac{1}{\lambda})^j\frac{\lambda}{\lambda-1}<1.$$

It suffices that $n(\log_\lambda(n)+1)\leqslant\lambda^j$, and therefore $\log_\lambda(n)+\log_\lambda(\log_\lambda(n)+1)\leqslant j$. Consequently, the expected number of invariant factors is

$$\sum_{j=1}^{\lceil\log_\lambda(n)+\log_\lambda(\log_\lambda(n)+1)\rceil} 1+\sum_{j=\lceil\log_\lambda(n)+\log_\lambda(\log_\lambda(n)+1)\rceil+1}^{n}\big(\lambda^{-n}+(\frac{1}{\lambda})^{j-1}\frac{\lambda}{\lambda-1}+3(\frac{2}{3})^j+$$
$$+2(\frac{1}{2})^j+\frac{n-j+1}{2}(\log_\lambda(n-j+1)+2))(\frac{1}{\lambda})^j\frac{\lambda}{\lambda-1}\big)=\lceil\log_\lambda(n)+\log_\lambda(\log_\lambda(n)+1)\rceil+$$
$$+1+\frac{\lambda+1}{(\lambda-1)^2}+4+1\leqslant\log_\lambda(n)+\log_\lambda(\log_\lambda(n)+1)+9.$$

□

B Modular determinant

Lemma B.1. *For $n\times n$ matrix A with entries chosen randomly and uniformly from the set $\{0,1\ldots S-1\}$, the probability that $p^l<S$ divides the determinant* $\det(A)$ *is at most $\frac{3}{p^l}$ for $p\neq 2,3,5$ and $\frac{4}{p^l}$ for $p=2,3,5$ and $S\geqslant 81$.*

Proof. To check whether $\mathrm{ord}_p(\det(A))\geqslant l$ we will consider a process of diagonalization for A mod p^l as described in Algorithm *LRE* of [5]. It consists of diagonalization and reduction steps. At ith diagonalization step, if an invertible entry is found, it is placed in the (i,i) pivot position and the ith row and column are zeroed. If no invertible entry is found, we proceed with a reduction step i.e. we consider the remaining $(n-i+1,n-i+1)$ minor divided by p. The problem now reduces to determining whether ord_p of an $(n-i+1,n-i+1)$ matrix is greater than $l-n+i-1$.
In the probabilistic consideration we need to determine the distribution of entries mod p^i after each reduction step. First, for A with entries chosen uniformly and randomly from the set $\{0,1\ldots S\}$, the probability that an entry is determined mod $p^i, i\leqslant l$, is less than or equal to $\beta_i(0)=\frac{1}{S}\lceil\frac{S}{p^i}\rceil$. After k reductions and m diagonalization steps we consider $i\leqslant l-2k$ (each reduction is performed on a matrix of order at least 2 and reduces the determinant by at least p^2) and a conditional probability of choosing the entries of a matrix A_k^m determined mod p^i with a probability at most $\beta_i(k)=\frac{1}{N_k}\lceil\frac{N_k}{p^i}\rceil$, where $N_k=\left\lceil\frac{\lceil\frac{S}{p}\rceil}{p}\right\rceil$ (the division is repeated k times). Since k is less than or equal to $\lceil l/2\rceil-1$ and $l\leqslant log_p(S)$, we have $N_k\geqslant\frac{S}{p^k}\geqslant\sqrt{S}$ and $\beta_i\leqslant\frac{2}{p^i}$ throughout the diagonalization process.

We will estimate the probability inductively. The estimations are performed for $p > 5$. First, for $l = 1$, we recall the result of [20, p.62] that

$$P(p \mid \det(A)) \leqslant \sum_{i=1}^{n} \beta_1 \leqslant \frac{3}{p}.$$

Then for $n = 2, l = 2$,

$$P(p^2 \mid \det(A)) \leqslant (1 - P(p \mid a_{ij} \forall_{i,j}))\beta_2 + P(p \mid a_{ij} \forall_{i,j}) \leqslant \beta_2 + (\beta_1)^{2^2} \leqslant \frac{3}{p^2}.$$

Again for $n = 2, 1 < l < n$ this becomes

$$P(p^l \mid \det(A)) \leqslant (1 - P(p \mid a_{ij} \forall_{i,j}))\beta_l + P(p \mid a_{ij} \forall_{i,j}) P(p^{l-2} \mid \det(A_1)).$$

Notice, that we sum over all possible diagonalization/reduction steps combinations. β_i bounds the probability of choosing the last diagonal entry determined mod p^i. A_1 is equal to A/p. By induction for $p > 5$, $P(p^l \mid \det(A))$ can be bounded by $\frac{2}{p^l} + (\frac{2}{p})^4 \frac{3}{p^{l-2}} \leqslant \frac{3}{p^l}$.
Now we will consider $n > 2$. Again we can sum over all possible diagonalization/reduction steps combinations and the resulting bound for the probability is

$$\begin{aligned} P(p^l \mid \det(A)) \leqslant & \sum_{i=n}^{l} (1 - P(p \mid a_{ij} \forall_{i,j \leqslant n})) \dots (1 - P(p \mid a_{ij}^{n-i+1} \forall_{i,j \leqslant i+1})) \beta_1^{i^2} \\ & + \sum_{i=l-1}^{2} (1 - P(p \mid a_{ij} \forall_{i,j \leqslant n})) \dots (1 - P(p \mid a_{ij}^{n-i+1} \forall_{i,j \leqslant i+1})) \beta_1^{i^2} P(p^{l-i} \mid \det(A_1^i)) \\ & + (1 - P(p \mid a_{ij} \forall_{i,j \leqslant n})) \dots (1 - P(p \mid a_{ij}^{n-1} \forall_{i,j \leqslant 2})) \beta_l \end{aligned}$$

for $l \leqslant n$ and similarly for $l > n$

$$\begin{aligned} P(p^l, A) \leqslant & \sum_{i=n}^{2} (1 - P(p \mid a_{ij} \forall_{i,j \leqslant n})) \dots (1 - P(p \mid a_{ij}^{n-i+1} \forall_{i,j \leqslant i+1})) \beta_1^{i^2} P(p^{l-i} \mid \det(A_1^i)) \\ & + (1 - P(p \mid a_{ij}^{n-i+1} \forall_{i,j \leqslant n})) \dots (1 - P(p \mid a_{ij}^{n-1} \forall_{i,j \leqslant 2})) \beta_l. \end{aligned}$$

Again, we can use the induction to get

$$P(p^l \mid \det(A)) \leqslant \big(\sum_{i=2}^{l} (\frac{2}{p})^{i^2} \frac{3}{p^{l-i}} \big) + \frac{2}{p^l} \leqslant \sum_{i=0}^{\infty} (\frac{3 \cdot 2^4}{p^{l+2}} (\frac{2^5}{p^4})^i) + \frac{2}{p^l} \leqslant \frac{3 \cdot 2^4 p^4}{p^{l+2}(p^4 - 2^5)} + \frac{2}{p^l} \leqslant \frac{3}{p^l}$$

for $p > 5$. For $p = 2, 3, 5$ by similar calculations we can prove that, provided that $N_k \geqslant 9$, $P(p^l, A) \leqslant \frac{4}{p^l}$. The condition on N_k is satisfied as soon as $\sqrt{S} \geqslant 9$ i.e. $S \geqslant 81$.

□

Laboratoire de Modélisation et Calcul
Université Joseph Fourier, Grenoble I
BP 53X, 38041 Grenoble, FRANCE
{Jean-Guillaume.Dumas;Anna.Urbanska}@imag.fr
www-lmc.imag.fr/lmc-mosaic/{Jean-Guillaume.Dumas;Anna.Urbanska}

Newton's method for the common eigenvector problem

Abdellatif El Ghazi Said El Hajji Luc Giraud Serge Gratton

Abstract

In [1] we have proved the sensitivity of computing the common eigenvector of two matrices, and we've designed a new approach to solve this problem based on the notion of the backward error.

In this paper we will use Newton's method to compute a common eigenvector for two matrices, taking the backward error as a stopping criteria, note that a recent optimization-based approach has been proposed for eigenvalue and generalized eigenvalue computations [3].

We mention that no assumptions are made on the matrices A and B.

Introduction

Let A and B be complex $n \times n$ matrices, a nonzero complex vector x is a common eigenvector of A and B if there exist complex numbers α and β such that:

$$\begin{cases} Ax = \alpha x \\ Bx = \beta x \end{cases} \qquad (P)$$

It's known that whenever the matrices A and B commute, they has at least one common eigenvector. Also if the matrices A and B can be simultaneously brought to an upper triangular form i.e : if there exists a nonsingular matrix P and triangular matrices R and S such that:

$$\begin{cases} P^{-1}AP = R \\ P^{-1}BP = S \end{cases}$$

Then the first column of P is a common eigenvector of A and B.

In 1984 D. Shemesh [2] gave a computable criterion for two square matrices to possess a common eigenvector:

Theorem 0.1 (Dan Shemesh). *Two square matrices A and B have a common eigenvector if and only if*

$$\mathcal{L} = \bigcap_{k,l=1}^{n-1} Ker[A^k, B^l] \neq \{0\}$$

The Shemesh theoretical condition is not easy to bring into use in practice for two reasons:

First, it relies on an algorithm to compute the powers of matrices A and B, (the sensitivity of computing the matrix powers is detailed by N.Higham in [4]), then compute the intersection of the null-space of the matrices $[A^k, B^l]$. These algorithms are closely related to the problem of the rank determination, that involves a lot of thresholds, whose determination is delicate and crucial to ensure the backward stability of the algorithm,(more about stability can be found in [6, 7]).

Second, the condition enables to decide when a common eigenvector exists, but does not provide an algorithm for computing it.

So rise the necessity to build a numerical algorithm to compute the common eigenvector when it exist.

1 Ill-poseness of the problem

In [1] we've shown that it is not easy to find a common eigenvector of a pair of matrices in the presence of round-off errors since the related perturbations are likely to transform a pair of matrices having a common eigenvector into a pair which does not enjoy this property.

We've proved the topological reason beyond this in the theorem :

Theorem 1.1. *[1]:*
The set of matrices which does not have any eigenvector in common is dense in the set of all pairs of matrices i.e.

$$\bar{S} = M_n(\mathbb{C})^2$$

where

$$S = \{(A, B) \in M_n(\mathbb{C})^2 / A \text{ and } B \text{ does not have a common eigenvector}\}$$

Then we've defined a new concept of approximate common eigenvector based on the notion of the backward error defined by :

Definition 1.2. Let $\tilde{x}$ be an approximation of the solution of the problem (P), the backward error η associated with $\tilde{x}$, noted $\eta(\tilde{x})$, is given by:

$$\eta(\tilde{x}) = \min_{\tilde{\alpha},\tilde{\beta}} \min\{\epsilon : \left[\begin{array}{l} (A + \Delta A)\tilde{x} = \tilde{\alpha}\tilde{x} \\ (B + \Delta B)\tilde{x} = \tilde{\beta}\tilde{x} \end{array} \right. \}$$

where

$$\epsilon = \sqrt{\frac{\|\Delta A\|^2}{\|A\|^2} + \frac{\|\Delta B\|^2}{\|B\|^2}}.$$

Furthermore, we have proved an explicit expression of the backward error $\eta(\tilde{x})$.

Theorem 1.3. *[1] The backward error is given by:*

$$\eta(\tilde{x}) = \sqrt{\frac{\|A\tilde{x} - \frac{(\tilde{x}^T A\tilde{x})}{\|\tilde{x}\|^2}\tilde{x}\|^2}{\|A\|^2\|\tilde{x}\|^2} + \frac{\|B\tilde{x} - \frac{(\tilde{x}^T B\tilde{x})}{\|\tilde{x}\|^2}\tilde{x}\|^2}{\|B\|^2\|\tilde{x}\|^2}}.$$

In the next section, we will use Newton's method to find an approximation to the common eigenvector , we'll use the backward error as a stopping criteria.

2 Gauss-Newton's method.

$\bar{x}$ is a common eigenvector of A and B if

$$\begin{cases} A\bar{x} = \alpha\bar{x} \\ B\bar{x} = \beta\bar{x} \end{cases}$$

Since an eigenvector is not unique, we can suppose that $\bar{x}$ verify for a given y :

$$y^T\bar{x} = 1$$

then $\alpha = y^TA\bar{x}$ and $\beta = y^TB\bar{x}$

Set

$$\begin{array}{rccc} F: & \mathbb{C}^n & \longrightarrow & \mathbb{C}^{2n+1} \\ & x & \longmapsto & \begin{bmatrix} Ax - (y^TAx)x \\ Bx - (y^TBx)x \\ y^Tx - 1 \end{bmatrix} \end{array}$$

We have $F(\bar{x}) = 0$.

In the neighborhood of the current iterate x_c, F can be expanded in Taylor series

$$F(x_c + p) = F(x_c) + J(x_c).p + O(p^2)$$

J is the gradient of F.

$$J(x) = \begin{bmatrix} (I - xy^T)A - (y^TAx)I \\ (I - xy^T)B - (y^TBx)I \\ y^T \end{bmatrix}$$

By neglecting terms of order p^2 and higher and choosing a step p such that

$$F(x_c + p) = 0$$

we obtain $J(x_c).p = -F(x_c)$, the correction p are then added to the solution, we get the algorithm:

step 0 : choose x_0
step 1 : repeat until convergence
  step 1.1 : solve $J(x_k)p_k = -F(x_k)$**.**
  step 1.2 : update $x_{k+1} = x_k + p_k$

Newton's method is attractive because under appropriate conditions it converges rapidly from any sufficiently good initial guess. In particular, if the Jacobian is nonsingular at the solution, local quadratic convergence can be proved [5,Theorem 5.2.1 page 90]. The Kantorovich Theorem yields a weaker bound on the convergence rate but makes no assumption on the nonsingularity of Jacobian at the solution [5,Theorem 5.3.1 page 92].

2.1 The damped Gauss-Newton.(DGN)

To accept the Newton step p we require that it decrease $\|F\|$, this is the same requirement we would impose if we were trying to minimize

$$f = \frac{1}{2}F^T.F = \frac{1}{2}\|F\|^2$$

To get a more useful method we take instead

$$x_{k+1} = x_k + \lambda_k p_k$$

Where λ_k is a parameter to be fixed so $f(x_k + \lambda_k p_k)$ decrease sufficiently.

The Newton step p is a descent direction for f since:

$$\nabla f^T.p = F^T.J.p = -F^T.F < 0$$

The resulting method, is called the damped Gauss-Newton method.

2.1.1 Backtracking:

The common procedure of choosing λ_k is to try the full Newton step i.e choose $\lambda_k = 1$ because once we are close enough to the solution we will get quadratic convergence. However, we check at each iteration that the proposed step reduces f. If not, we backtrack along the Newton direction until we have an acceptable step. Because the Newton step is a descent direction for f, we are guaranteed to find an acceptable step by backtracking.

The criterion $f(x_{k+1}) < f(x_k)$ can fail to converge in one of two ways:

- f is decreasing too slowly relative to the step lengths
- The steps are too small, relative to the initial rate of decrease of f

To fix the first problem we require that the average rate of decrease of f to be at lest some fraction $a \in]0, 1[$ of the initial rate of decrease in that direction:

$$f(x_{k+1}) \leq f(x_k) + a\nabla f^T.(x_{k+1} - x_k) \quad (1)$$

The second problem can be fixed by requiring the rate of decrease of f at x_{k+1} be larger than some fraction $b \in]a, 1[$ of the rate of decrease of f at x_k:

$$\nabla f(x_k)^T.(x_{k+1} - x_k) \geq b\nabla f(x_{k+1})^T.(x_{k+1} - x_k) \quad (2)$$

Then we choose for λ_k the largest number in the sequence 1, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$.... such that (1) and (2) hold. In practice we haven't to check condition (2) because backtracking avoid excessively small steps. We get the algorithm:

step 0 : choose x_0 **and set** $x_0 = \frac{x_0}{y^T.x_0}$**, choose** $a \in]0, 1[$

step 1 : $\lambda_k = 1$ **and repeat until convergence**
step 1.1 : solve $J(x_c)p_c = -F(x_c)$
step 1.2 : update $x_{k+1} = x_k + \lambda_k p_k$**.**
step 1.3 : If (1) **holds set** $x_k = x_{k+1}$**, and go to step 1 else** $\lambda_k = \frac{1}{2}\lambda_k$**, and go to step 1.1**

2.1.2 Exact Line searches

We can choose λ_k so that x_{k+1} exactly minimizes f in the direction p_k:

$$\lambda_k = \min_{\lambda} f(x_k + \lambda p_k)$$

or

$$f(x + \lambda p) = \frac{1}{2}\|F(x + \lambda p)\|^2$$

with

$$F(x + \lambda p) = \begin{bmatrix} A(x + \lambda p) - (y^T A(x + \lambda p))(x + \lambda p) \\ B(x + \lambda p) - (y^T B(x + \lambda p))(x + \lambda p) \\ y^T(x + \lambda p) - 1 \end{bmatrix}$$

Let

$$\begin{aligned} u_1 &= -(y^T Ap)p \\ u_2 &= -(y^T Bp)p \\ v_1 &= Ap - (y^T Ax)p - (y^T Ap)x \\ v_2 &= Bp - (y^T Bx)p - (y^T Bp)x \\ v_3 &= y^T p \\ w_1 &= Ax - (y^T Ax)x \\ w_2 &= Bx - (y^T Bx)x \\ w_3 &= y^T x - 1 \end{aligned}$$

then

$$F(x + \lambda p) = \begin{bmatrix} u_1\lambda^2 + v_1\lambda + w_1 \\ u_2\lambda^2 + v_2\lambda + w_2 \\ v_3\lambda + w_3 \end{bmatrix}$$

Let

$$\begin{aligned} a_4 &= \tfrac{1}{2}(\|u_1\|^2 + \|u_2\|^2) \\ a_3 &= Re(u_1^T v_1) + Re(u_2^T v_2) \\ a_2 &= \tfrac{1}{2}(\|v_1\|^2 + \|v_2\|^2 + 2Re(u_1^T w_1) + 2Re(u_2^T w_2) + v_3^2) \\ a_1 &= Re(v_1^T w_1) + Re(v_2^T w_2) + w_3 v_3 \\ a_0 &= \tfrac{1}{2}(\|w_1\|^2 + \|w_2\|^2 + w_3^2) \end{aligned}$$

then

$$f(x + \lambda p) = a_4\lambda^4 + a_3\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0$$

and

$$\frac{df}{d\lambda}(x + \lambda p) = 4a_4\lambda^3 + 3a_3\lambda^2 + 2a_2\lambda + a_1$$

It's root are the eigenvalues of the companion matrix:

$$C = \begin{pmatrix} 0 & 0 & -\frac{1}{4}\frac{a_1}{a_4} \\ 1 & 0 & -\frac{1}{2}\frac{a_2}{a_4} \\ 0 & 1 & \frac{3}{4}\frac{a_3}{a_4} \end{pmatrix}$$

Then λ_k is the eigenvalue of C that minimize f. We get the algorithm:

step 0 : choose x_0 **and set** $x_0 = \frac{x_0}{y^T.x_0}$**,**
step 1 : repeat until convergence
step 1.1 : solve $J(x_k)p_k = -F(x_k)$
step 1.2 : compute a_1, a_2, a_3, a_4**, and the matrix** C**.**
step 1.3 : compute the eigenvalues of C**, and** λ_k
step 1.4 : update $x_{k+1} = x_k + \lambda_k p_k$**.**

3 Newton's method.

As the function f is special and easy to have it's second derivative, we can use (directly) the Newton's method, and take

$$f(x_c + p) \simeq f(x_c) + \nabla f(x_c)^T p + \frac{1}{2} p^T \nabla^2 f(x_c) p$$

Since $f = \frac{1}{2} F^T(x) F(x)$

$$\nabla f = J(x)^T F(x)$$

and

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{i=1}^{2n+1} F_i(x) \nabla^2 F_i(x)$$

where

$$F_i(x) = e_i^T F(x),\ e_i = (0, ..., 0, 1, 0, ..., 0)^T$$

and the minimizer of f is given by

$$x_{k+1} = x_k - [J(x_k)^T J(x_k) + \sum_{i=1}^{2n+1} F_i(x) \nabla^2 F_i(x_k)]^{-1} J(x)^T F(x_k)$$

with

$$\sum_{i=1}^{2n+1} F_i(x) \nabla^2 F_i(x) = \sum_{i=1}^{2n+1} (e_i^T F(x))(e_i^T \nabla^2 F(x)) =$$

$$\sum_{i=1}^{n} e_i^T (Ax - (y^T Ax)x).e_i^T \nabla^2 (Ax - (y^T Ax)x) + \sum_{j=1}^{n} e_j^T (Bx - (y^T Bx)x) e_i^T \nabla^2 (Bx - (y^T Bx)x) +$$

$$(y^T x - 1) \nabla^2 (y^T x - 1)$$

Or

$$e_i^T\nabla^2(Ax-(y^TAx)x)=e_i^T\nabla(A-(y^TAx)I-xy^TA)=e_i^T\nabla(-(y^TAx)I-xy^TA)=$$
$$\nabla(-(y^TAx)e_i^T-e_i^Txy^TA)=\nabla(-x^TA^Tye_i^T-x^Te_iy^TA)=-A^Tye_i^T-e_iy^TA$$

Then

$$\sum_{i=1}^{n}e_i^T(Ax-(y^TAx)x).e_i^T\nabla^2(Ax-(y^TAx)x)=\sum_{i=1}^{n}e_i^T(Ax-(y^TAx)x)(-A^Tye_i^T-e_iy^TA)=$$

$$-\sum_{i=1}^{n}[A^Tye_i^T(Ax-(y^TAx)x)e_i^T+e_i^Te_i(Ax-(y^TAx)x)y^TA]=$$

$$-A^Ty(Ax-(y^TAx)x)^T-(Ax-(y^TAx)x)y^TA$$

And

$$(y^Tx-1)\nabla^2(y^Tx-1)=0$$

We get

$$\sum_{i=1}^{2n+1}F_i(x)\nabla^2F_i(x)$$

$$=-(A^Ty(Ax-(y^TAx)x)^T+(Ax-(y^TAx)x)y^TA+B^Ty(Bx-(y^TBx)x)^T+(Bx-(y^TBx)x)y^TB)$$

and we get the algorithm :

step 0 : choose x_0 **and set** $x_0=\frac{x_0}{y^T.x_0}$**,**
step 1 : repeat until convergence
step 1.1 : compute $J(x_k)$**,** $\alpha_k=y^TAx_k$ **and** $\beta_k=y^TBx$
step 1.2 : compute $e_A=Ax_k-\alpha_kx_k$**,**$e_B=Bx_k-\beta_kx_k$**, and set**

$$Df(x_k)=[J^TJ-A^Tye_A^T+e_Ay^TA+B^Tye_B^T+e_By^TB]$$

step 1.3 : compute $p=DF^{-1}J^TF$**, and update** $x_{k+1}=x_k+p$**.**

3.1 Test examples

For a given vector u, we build a couple of matrices A and B having u as a common eigenvector. To avoid to have a common eigenvector with multiplicity 1, we'll associate the common eigenvector u with a Jordan block.

- $J_A=\begin{pmatrix}\overbrace{\begin{bmatrix}\alpha&1&0&\cdots&0\\0&\alpha&1&&0\\\vdots&&\ddots&\ddots&\\0&&&\alpha&1\\0&&&0&\alpha\end{bmatrix}}^{Jordan\ block}&&0\\&0&&&\text{[Random matrix]}\end{pmatrix}$

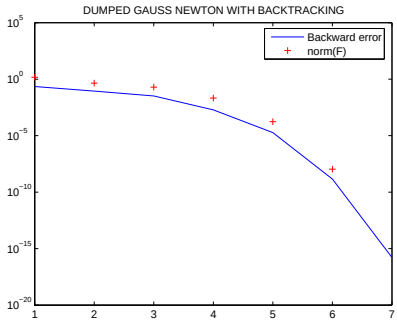
- With the same technique with different parameter we build J_B
- Complete u to build a basis of $\mathbb{R}^n$ and set Q_A the corresponding matrix.
- Set $v = (1, 2, ...n)^T$ or any vector linearly independent with u, and complete $< u, v >$ to get a basis for $\mathbb{R}^n$ and set Q_B the corresponding matrix.
- Set $A = Q_A.J_A.Q_A^T$.
- Set$B = Q_B.J_B.Q_B^T$.

Then A and B have u as a common eigenvector.

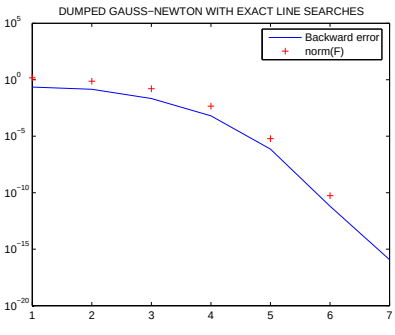
3.1.1 Numerical experiments

We note by e_i the i^{th} element of the canonical basis of $\mathbb{R}^n$ and $x = rand(n, 1)'$ is a Matlab notation to design a randomize vector of size n.
The example below summarize what we've got for many test with different matrices and different sizes. we generate two square 20×20 matrices with a common eigenvector $u = (1, 1..., 1)^T$ each component of u is equal to one. with an inial guess $x = rand(n, 1)'$ we found:

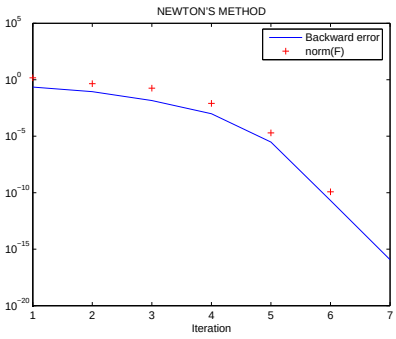
	DGN with backtraking	DGN with exact line search	Newton
iteration	7	6	6
the approximation	u	u	u



DGN with backtraking



DGN with exact line search



Newton

References

[1] A. El Ghazi, S. El Hajji, S. Graton, L. Giraud, *Backward error for the common eigenvector problem*, Rapport Interne, soumis.

[2] D. Shemesh, *Commun eigen vectors of two matrices*, Linear Algebra and its Applications 62, (1984), pp. 11-18.

[3] K. Wu, Y. Saad, and A. Stathopoulos, *Inexact Newton preconditioning techniques for eigenvalue problems*, Electronic Transactions on Numerical Analysis, 7 (1998), pp. 202.

[4] N. J.Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 2002, pp.339–353.

[5] J.E.Dennis, Jr. and R.B. Scnabel., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* , Prentice-Hall, Englewood Cliffs, NJ, USA, (1983).

[6] F. Chaitin-Chatelin and V. Fraysse,*Lectures on Finite Precision Computations*, SIAM,(1996),pp.53–73.

[7] G. Stewart, *Determining rank in the presence of error*, NATO ASI Series, Series E: Applied Sciences, Vol. 232, ed. M. S. Moonen, G. H. Golub and B. L. R. De Moor, Kluwer, (1993), pp. 275–291.

Abdellatif EL GHAZI
Faculté des sciences
Université Med V Agdal Rabat Maroc. aelghazi@msn.com
http://www.fsr.ac.ma/ANO/elghazi.htm

Said EL HAJJI
Faculté des sciences
Université Med V Agdal Rabat Maroc
elhajji@fsr.ac.ma
http://www.fsr.ac.ma/ANO/elhajji.htm

Luc Giraud
CERFACS
42 av. Gaspard Coriolis, 31057 Toulouse Cedex, France.
giraud@cerfacs.fr
http://www.enseeiht.fr/ giraud/

Serge Gratton
CERFACS
42 av. Gaspard Coriolis, 31057 Toulouse Cedex, France.
gratton@cerfacs.fr
http://www.cerfacs.fr/ gratton/gratton.html

Latin squares associated to principal autotopisms of long cycles. Application in Cryptography

Raúl M. Falcón Ganfornina

Abstract

Fixed a principal isotopism $\Theta = (\alpha, \beta, \epsilon) \in S_n^3$, where S_n is the symmetric group of the set $N = \{0, 1, ..., n-1\}$, we are going to study in this paper the number $\Delta(\Theta)$ of Latin squares which have Θ as a principal autotopism. As an application in Cryptography, we use it in the construction of secret sharing schemes based in $\mathfrak{F}$-critical sets of Latin squares.

Keyword: Latin square, Autotopism group, Critical set, Secret sharing scheme.

1 Introduction

A *quasigroup* [1] is a nonempty set G endowed with a product $\cdot$, such that if any two of the three symbols a, b, c in the equation $a \cdot b = c$ are given as elements of G, the third is uniquely determined as an element of G. It is equivalent to say that G is endowed with left / and right \ division. Two quasigroups $(G, \cdot)$ and $(H, \circ)$ are *isotopic* [4] if there are three bijections α, β, γ from H to G, such that:

$$\gamma(a \circ b) = \alpha(a) \cdot \beta(b), \text{ for all } a, b \in H.$$

The triple $\Theta = (\alpha, \beta, \gamma)$ is called an *isotopism* from $(G, \cdot)$ to $(H, \circ)$. If $G = H$ and $\alpha = \beta = \gamma$, the isotopism is indeed an *isomorphism*. If $\gamma = \epsilon$, the identity map on G, Θ is called a *principal isotopism*. If $G = H$ and $\cdot \equiv \circ$, Θ is called an *autotopism*. Finally, $\Theta = (\epsilon, \epsilon, \epsilon)$ is called the *trivial autotopism*.

If we consider the multiplication table of a quasigroup, we obtain a Latin square. A *Latin square*, L, of order n, is a $n \times n$ array with elements chosen from a set of n symbols $N = \{x_1, ..., x_n\}$, such that each symbol occurs precisely once in each row and each column. A *Latin subrectangle* of L is a rectangular subarray R of L such that exactly the same symbols occur in each row of R. The set of Latin squares of order n is denoted by $LS(n)$. A *partial Latin square*, P, of order n, is a $n \times n$ array with elements chosen from a set of n symbols, such that each symbol occurs at most once in each row and in each column. The set of partial Latin squares of order n is denoted as $PLS(n)$. It is said that a fixed $P \in PLS(n)$ can be *uniquely completed* to a Latin square $L \in LS(n)$ if L is the unique Latin square such that $P \subseteq L$ and it is denoted $P \in UC(L)$. If besides any proper subset

of P can be completed to two distinct Latin squares it is said that P is a *critical set* of L and it is denoted $P \in CS(L)$. A critical set of L is said *minimal* if it has the smallest size of all possible critical sets of L. Critical sets were introduced in the last 70's of the past century [15], [6]. Applications of them in Cryptography were obtained by Seberry [18] in 1990. Later on, it has been proved that critical sets allow to construct secret sharing schemes [5]. In [10] it can be observed some of these applications to Cryptography.

The cardinality of $LS(n)$ for all $n \in \mathbb{N}$, $N(n,n)$, is still an open problem, although it is known that this cardinality grows exponentially. Studies of $N(n,n)$ with $n \leq 11$ can be found in [20], [2] or [14]. We will consider from now on $N = \{0, 1, ..., n-1\}$. So, if $L = (l_{ij})$, the *orthogonal array representation of* L is the set of n^2 triples $\{(i, j, l_{ij}) : 0 \leq i, j \leq n-1\}$. An *isotopism* of a Latin square L is a triple $\Theta = (\alpha, \beta, \gamma) \in \mathcal{I}_n = S_n \times S_n \times S_n$, where S_n is the symmetric group on N and so, α, β and γ are respectively, permutations of rows, columns and symbols of L. The resulting square L^Θ is also a Latin square and it is said to be *isotopic* to L. In particular, if $L = (l_{ij})$, then $L^\Theta = \{(i, j, \gamma^{-1}\left(l_{\alpha(i)\beta(j)}\right) : 0 \leq i, j \leq n-1\}$. The set of all Latin squares isotopic to L is called its *isotopy class*. An isotopism which maps L to itself is an *autotopism*. The stabilizer subgroup of L in $\mathcal{I}_n$ is its *autotopism group*, $\mathcal{U}(L) = \{\Theta \in \mathcal{I}_n : L^\Theta = L\}$. Given $P \in PLS(n)$, contained in L, and $\mathfrak{F} \subseteq \mathcal{U}(L)$, it is defined the *extended autotopy* $P^{\mathfrak{F}} = \bigcup_{\Theta \in \mathfrak{F}} P^\Theta \in PLS(n)$.

Cardinalities of isotopy classes and autotopism groups have been already studied, for example in [17], [7] or, more recently, in [13] and [14]. In these two last papers, authors have used autotopism group sizes (computed by B.D. McKay's *nauty* [11]) to give counts of Latin squares of order up to 11. Indeed, as a first step to obtain it, they have studied the possible autotopisms of a given Latin square. To do it, they have defined the *cycle structure of a permutation* γ as the sequence $(n_1, n_2, ...)$, where n_i is the number of cycles of length i in γ. So, they have proved the following:

Theorem 1.1. (McKay, Meynert and Myrvold [13]) *Let $L \in LS(n)$. Every non-trivial $\Theta = (\alpha, \beta, \gamma) \in \mathcal{U}(L)$ verifies one of the following assertions:*

a) α, β, γ have the same cycle structure with at least one and at most $\lfloor n/2 \rfloor$ fixed points,

b) One of α, β, γ has at least one fixed point and the other two have the same cycle structure without fixed points,

c) None of α, β, γ has fixed points. □

Also in these papers, they have studied the reciprocal question. That is, given an isotopism $\Theta = (\alpha, \beta, \gamma) \in \mathcal{I}_n$, how many Latin squares there exist such that Θ is an autotopism of all of them. However, they are not interested in the number of Latin squares but in the number of isotopy classes. Besides, they only study [13] some concrete cases of autotopisms:

a) For some prime p, α, β and γ have order p with the same number m of fixed points, where $1 \leq m \leq \lfloor n/2 \rfloor$.

b) For some prime p dividing n, α and β have order p and no fixed points, and γ has order 1 or p. If $p = 2$ and $n \equiv 2$ $(mod\ 4)$, γ has at least two fixed points.

To obtain the previous number, they use computer programs which incorporate two methods of approach to generation: the orderly approach method [9], [16] and the canonical construction path method [12]. In particular, this last one allows to construct a Latin square one *row block* at a time, where a row block consists of the rows which correspond to a cycle of α.

Nevertheless, a study of the number of Latin squares associated to any autotopism is even necessary. Indeed, this question will allow to study better the problem of the smallest size of $\mathfrak{F}$-critical sets [8]: Fixed $L \in LS(n)$, $P \in PLS(n)$ contained in L and $\mathfrak{F} \subseteq \mathcal{U}(L)$, it is defined $\mathfrak{F}(P) = P^{<\mathfrak{F}>}$, where $<\mathfrak{F}>$ is the subgroup of $\mathcal{U}(L)$ generated by $\mathfrak{F}$. Then, P is *uniquely* $\mathfrak{F}$*-completable to* L, which is denoted as $P \in UC_{\mathfrak{F}}(L)$, if $\mathfrak{F}(P) \in UC(L)$. Furthermore, P is a $\mathfrak{F}$*-critical set* if $P \in UC_{\mathfrak{F}}(L)$ and $Q \notin UC_{\mathfrak{F}}(L)$ for all $Q \subset P$. Analogous to critical sets, it is expected that $\mathfrak{F}$-critical sets will have applications in Cryptography, specially as secret sharing schemes.

In this paper we will start this study with a particular case of autotopisms, the principal ones, which have been partially studied in [13], although only to get the number of isotopy class. The paper is structured as follows. In the next section, fixed a principal isotopism $\Theta = (\alpha, \beta, \epsilon) \in \mathcal{I}_n$, we will study the number $\Delta(\Theta)$ of Latin squares which have Θ as a principal autotopism. First, we will prove that α and β must have the same cycle structure with all their cycles of the same length and without fixed points. Then, we will study the cases in which this length is $\frac{n}{k}$, with $k \in \{1, 2, 3, 4\}$. We will use the canonical construction path to generate the associated Latin squares. So, in the general case, we will see that:

$$\Delta(\Theta) = n! \cdot \left(\frac{n}{k}!\right)^{k(k-1)} \cdot \Omega(\Theta),$$

where $\Omega(\Theta)$ is the number of different ways in which we can choose a determined set of row blocks. Finally, the paper finishes in the third section with a study in Cryptography about the possible use of a set $\mathfrak{F}$ of autotopisms of a Latin square L as shares of a secret sharing scheme. To get it, we will keep in mind the concept of $\mathfrak{F}$-critical set of L.

2 Principal autotopisms of Latin squares

Fixed $n \in \mathbb{N}$ and $\Theta \in \mathcal{I}_n$, we will denote by $\Delta(\Theta)$ the number of Latin squares of order n such that Θ is an autotopism of all of them, and by $LS(\Theta)$ the set of such Latin squares. That is, $\Delta(\Theta) = |LS(\Theta)|$ and $L \in LS(\Theta)$ if and only if $\Theta \in \mathcal{U}(L)$. In this paper, we are interested in the value of $\Delta(\Theta)$ if Θ is a principal isotopism, that is, if $\Theta = (\alpha, \beta, \epsilon)$, where ϵ is the identity map in $N = \{0, 1, ..., n-1\}$. It is clear that $(\epsilon, \epsilon, \epsilon) \in \mathcal{U}(L)$ for all $L \in LS(n)$. So, $\Delta((\epsilon, \epsilon, \epsilon)) = N(n, n)$, the number of Latin squares of orden n. Therefore, we must study when Θ is a non-trivial principal autotopism of a Latin square.

Let us see a result which allows to fix the structure of Θ in the more general case in which ϵ is one of the permutations of Θ:

Proposition 2.1. *Let $\Theta = (\alpha, \beta, \gamma) \in \mathcal{I}_n$ be a non-trivial isotopism. If one of the permutations α, β or γ is equal to ϵ, then $\Delta(\Theta) > 0$ only if the other two permutations have the same cycle structure with all their cycles of the same length and without fixed points.*

Proof.

We are in the case (b) of Theorem 1.1. So, if the other two permutations have not the same cycle structure or have fixed points, then $\Delta(\Theta) = 0$. Let us study the different possibilities:

a) If $\alpha = \epsilon$, let us consider that β and γ have the same cycle structure without fixed points. Let us take $b, c \in N = \{0, 1, ..., n-1\}$ such that b appears in a cycle of length λ_β of β, $(bx_2x_3...x_{\lambda_\beta})$, and c appears in a cycle of length λ_γ of γ, $(cy_2y_3...y_{\lambda_\gamma})$. We can suppose that $\lambda_\beta > \lambda_\gamma$. If $L = (l_{ij}) \in LS(n)$ is such that $\Theta \in \mathcal{U}(L)$, there must exist $a \in N$ such that $l_{ab} = c$. So, $l_{ab} = c = l_{ax_{\lambda_\gamma+1}}$, which is a contradiction with being $L \in LS(n)$.

b) If $\beta = \epsilon$, we reason analogously to (a).

c) If $\gamma = \epsilon$, let us consider that α and β have the same cycle structure without fixed points. Let us take $a, b \in N$ such that a appears in a cycle of length λ_α of α, $(ax_2x_3...x_{\lambda_\alpha})$, and b appears in a cycle of length λ_β of β, $(by_2y_3...y_{\lambda_\beta})$. We can suppose that $\lambda_\alpha < \lambda_\beta$. If $L = (l_{ij}) \in LS(n)$ is such that $\Theta \in \mathcal{U}(L)$, then $l_{ab} = l_{ay_{\lambda_\alpha+1}}$, which is a contradiction with being $L \in LS(n)$. □

Keeping in mind the previous proposition, we will be interested from now on in principal autotopisms $\Theta = (\alpha, \beta, \epsilon)$, such that α and β have the same cycle structure with all their cycles of the same length and without fixed points. Given such a Θ, we are interested in the exact value of $\Delta(\Theta)$. To see it, we start with cycles of length n and later on, we will decrease this length.

2.1 Cycles of length n

If α and β are both cycles of length n, we obtain the following result:

Proposition 2.2. *Let $\Theta = (\alpha, \beta, \epsilon) \in \mathcal{I}_n$ be such that α and β are both cycles of length n. Then, $\Delta(\Theta) = n!$.*

Proof.

Let $\alpha = (a_0a_1...a_{n-1})$ and $\beta = (b_0b_1...b_{n-1})$ be two cycles of length n of N. We can obtain a Latin square $L = (l_{ij})$ such that $\Theta \in \mathcal{U}(L)$. To do it, for all $i \in N$, let us take $l_{0i} \in N$, such that $l_{0j} \neq l_{0k}$ for all $j \neq k$. We can supppose that $a_0 = 0$. Now, fixed $i \in N$, we take $t_i \in N$ such that $b_{t_i} = i$. So, $l_{a_j b_{t_i+j \pmod n}} = l_{0i}$, for all $j \in N$. In this way, we can define the Latin square L. Furthermore, by swapping the elements l_{0i} in N, we can obtain $n!$ distinct Latin squares and it cannot exist other one such that has Θ as an autotopism. □

Let us see an example:

Example 2.3. Let us consider $n = 3$ and $N = \{0, 1, 2\}$. There are 36 elements of $\mathcal{I}_3$ with the form $(\alpha, \beta, \epsilon)$. However, from Proposition 2.1, only five of them are autotopisms of some Latin square of order 3. They are:

$$\Theta_1 = (\epsilon, \epsilon, \epsilon), \quad \Theta_2 = ((012), (012), \epsilon), \quad \Theta_3 = ((012), (021), \epsilon)$$
$$\Theta_4 = ((021), (012), \epsilon), \quad \Theta_5 = ((021), (021), \epsilon)$$

Besides, it can be seen that:

$$LS(\Theta_1) = LS(3)$$
$$LS(\Theta_2) = LS(\Theta_5) = \left\{ \begin{pmatrix} a & b & c \\ c & a & b \\ b & c & a \end{pmatrix} \in LS(3) : a, b, c \in N \right\}$$
$$LS(\Theta_3) = LS(\Theta_4) = \left\{ \begin{pmatrix} a & b & c \\ b & c & a \\ c & a & b \end{pmatrix} \in LS(3) : a, b, c \in N \right\}$$

So, $\Delta(\Theta_1) = 12$ and $\Delta(\Theta_i) = 6$, if $i \in \{2, 3, 4, 5\}$. Let us observe that $LS(\Theta_2) \cap LS(\Theta_3) = \emptyset$ and that $LS(\Theta_2) \cup LS(\Theta_3) = LS(3)$. $\lhd$

2.2 Cycles of length $\frac{n}{2}$

Now, if $n > 2$ is even and if α and β are both cycles of length $\frac{n}{2}$, we obtain the following result:

Proposition 2.4. *Let $\Theta = (\alpha, \beta, \epsilon) \in \mathcal{I}_n$, where $n > 2$ is even, be such that α and β are both the composition of two cycles of length $\frac{n}{2}$. Then, $\Delta(\Theta) = n! \cdot \left(\frac{n}{2}!\right)^2$.*

Proof.

Let us suppose that:

$$\alpha = (a_0 a_1 ... a_{\frac{n}{2}-1})(a_{\frac{n}{2}} a_{\frac{n}{2}+1} ... a_{n-1}), \qquad \beta = (b_0 b_1 ... b_{\frac{n}{2}-1})(b_{\frac{n}{2}} b_{\frac{n}{2}+1} ... b_{n-1}).$$

By using the canonical construction path [12], we can obtain a Latin square $L = (l_{ij})$ such that $\Theta \in \mathcal{U}(L)$. To do it, similarly to Proposition 2.2, we take $l_{a_0 i} \in N$ for all $i \in N$, such that $l_{a_0 j} \neq l_{a_0 k}$ for all $j \neq k$. Now, fixed $i \in N$, we take $t_i \in N$ such that $b_{t_i} = i$. So, $l_{a_j b_{t_i + j \pmod{\frac{n}{2}}}} = l_{a_0 i}$, for all $j \in \{0, 1, ..., \frac{n}{2} - 1\}$. In this way, we can define a Latin subrectangle R of L of $\frac{n}{2}$ rows and n columns. Indeed, R is a row block, because its rows correspond to the cycle $(a_0 a_1 ... a_{\frac{n}{2}-1})$. Besides, by swapping the elements l_{0i} in N, we can obtain $n!$ different Latin subrectangles of L, all of them associated by construction to the same rows.

Now, we do the same process with $a_{\frac{n}{2}}$ in the place of a_0, although when we choose the elements $l_{a_{\frac{n}{2}} i} \in N$, we must keep in mind R, as L must be a Latin square. That is, it must be $l_{a_{\frac{n}{2}} i} \in \{l_{a_0 \frac{n}{2}}, l_{a_0 (\frac{n}{2}+1)}, ..., l_{a_0 n}\}$ for all $i \in \{0, 1, ..., \frac{n}{2} - 1\}$ and $l_{a_{\frac{n}{2}} i} \in \{l_{a_0 1}, l_{a_0 2}, ..., l_{a_0 (\frac{n}{2}-1)}\}$

for all $i \in \{\frac{n}{2}, \frac{n}{2}+1, ..., n\}$. Therefore, in this way we can obtain finally $n! \cdot \left(\frac{n}{2}!\right)^2$ different Latin squares which have Θ as an autotopism. □

Let us see an example:

Example 2.5. Let us consider $n = 4$ and $N = \{0, 1, 2, 3\}$. If $\Theta = (\alpha, \beta, \epsilon) \in \mathcal{I}_4$ is a principal isotopy such that α and β are both products of two cycles of length 2, Θ must be one of the followings:

$$\Theta_1 = ((01)(23), (01)(23), \epsilon); \quad \Theta_2 = ((01)(23), (02)(13), \epsilon);$$
$$\Theta_3 = ((01)(23), (03)(12), \epsilon); \quad \Theta_4 = ((02)(13), (01)(23), \epsilon);$$
$$\Theta_5 = ((02)(13), (02)(13), \epsilon); \quad \Theta_6 = ((02)(13), (03)(12), \epsilon);$$
$$\Theta_7 = ((03)(12), (01)(23), \epsilon); \quad \Theta_8 = ((03)(12), (02)(13), \epsilon);$$
$$\Theta_9 = ((03)(12), (03)(12), \epsilon).$$

By swapping the values of a, b, c, d, e, f, g in N, we have that:

$LS(\Theta_1) = \left\{ \begin{pmatrix} a & b & c & d \\ b & a & d & c \\ e & f & g & h \\ f & e & h & g \end{pmatrix} \in LS(4) \right\}$	$LS(\Theta_2) = \left\{ \begin{pmatrix} a & b & c & d \\ c & d & a & b \\ e & f & g & h \\ g & h & e & f \end{pmatrix} \in LS(4) \right\}$
$LS(\Theta_3) = \left\{ \begin{pmatrix} a & b & c & d \\ d & c & b & a \\ e & f & g & h \\ h & g & f & e \end{pmatrix} \in LS(4) \right\}$	$LS(\Theta_4) = \left\{ \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ b & a & d & c \\ f & e & h & g \end{pmatrix} \in LS(4) \right\}$
$LS(\Theta_5) = \left\{ \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ c & d & a & b \\ g & h & e & f \end{pmatrix} \in LS(4) \right\}$	$LS(\Theta_6) = \left\{ \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ d & c & b & a \\ h & g & f & e \end{pmatrix} \in LS(4) \right\}$
$LS(\Theta_7) = \left\{ \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ f & e & h & g \\ b & a & d & c \end{pmatrix} \in LS(4) \right\}$	$LS(\Theta_8) = \left\{ \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ g & h & e & f \\ c & d & a & b \end{pmatrix} \in LS(4) \right\}$
$LS(\Theta_9) = \left\{ \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ h & g & f & e \\ d & c & b & a \end{pmatrix} \in LS(4) \right\}$	

So, $|LS(\Theta_i)| = \Delta(\Theta_i) = 4! \cdot (2!)^2 = 96$. By the other way, let us observe that $LS(\Theta_i) \cap LS(\Theta_j) = \emptyset$, except for:

(i,j)	$LS(\Theta_i) \cap LS(\Theta_j)$	(i,j)	$LS(\Theta_i) \cap LS(\Theta_j)$
$(1,5)$ $(1,9)$ $(5,9)$	$\left\{ \begin{pmatrix} a & b & c & d \\ b & a & d & c \\ c & d & a & b \\ d & c & b & a \end{pmatrix} \in LS(4) \right\}$	$(1,6)$ $(1,8)$ $(6,8)$	$\left\{ \begin{pmatrix} a & b & c & d \\ b & a & d & c \\ d & c & b & a \\ c & d & a & b \end{pmatrix} \in LS(4) \right\}$
$(2,4)$ $(2,9)$ $(4,9)$	$\left\{ \begin{pmatrix} a & b & c & d \\ c & d & a & b \\ b & a & d & c \\ d & c & b & a \end{pmatrix} \in LS(4) \right\}$	$(2,6)$ $(2,7)$ $(6,7)$	$\left\{ \begin{pmatrix} a & b & c & d \\ c & d & a & b \\ d & c & b & a \\ b & a & d & c \end{pmatrix} \in LS(4) \right\}$
$(3,4)$ $(3,8)$ $(4,8)$	$\left\{ \begin{pmatrix} a & b & c & d \\ d & c & b & a \\ b & a & d & c \\ c & d & a & b \end{pmatrix} \in LS(4) \right\}$	$(3,5)$ $(3,7)$ $(5,7)$	$\left\{ \begin{pmatrix} a & b & c & d \\ d & c & b & a \\ c & d & a & b \\ b & a & d & c \end{pmatrix} \in LS(4) \right\}$

Where $a, b, c, d \in N$. Therefore, as all the previous intersection contains 4! Latin squares and $\Delta(\Theta_i) = 4 \cdot 4!$ for all $i \in \{0, 1, ..., 9\}$, it can be seen that $\left|\bigcup_{i=1}^{9} LS(\Theta_i)\right| = 6 \cdot 4! + 9 \cdot 2 \cdot 4! = 24 \cdot 4! = 576 = |LS(4)|$. ◁

2.3 Cycles of length $\frac{n}{3}$

Let us suppose now that α and β are both cycles of length $\frac{n}{3}$:

Proposition 2.6. *Let $\Theta = (\alpha, \beta, \epsilon) \in \mathcal{I}_n$, where $n > 3$ is a multiple of 3, be such that α and β are both the composition of three cycles of length $\frac{n}{3}$. So:*

$$\Delta(\Theta) = n! \cdot \left(\frac{n}{3}!\right)^6 \cdot \sum_{k=0}^{n/3} \binom{n/3}{k}^3.$$

Proof.

Let us suppose that:

$$\alpha = (a_0 a_1 ... a_{\frac{n}{3}-1})(a_{\frac{n}{3}} a_{\frac{n}{3}+1} ... a_{\frac{2n}{3}-1})(a_{\frac{2n}{3}} a_{\frac{2n}{3}+1} ... a_{n-1}),$$

$$\beta = (b_0 b_1 ... b_{\frac{n}{3}-1})(b_{\frac{n}{3}} b_{\frac{n}{3}+1} ... b_{\frac{2n}{3}-1})(b_{\frac{2n}{3}} b_{\frac{2n}{3}+1} ... b_{n-1}).$$

To obtain a Latin square $L = (l_{ij})$ such that $\Theta \in \mathcal{U}(L)$, it is useful to consider the sets $S^{i,j} = \{l_{a_{i \cdot \frac{n}{3}} b_{j \cdot \frac{n}{3}}}, l_{a_{i \cdot \frac{n}{3}} b_{j \cdot \frac{n}{3}+1}}, ..., l_{a_{i \cdot \frac{n}{3}} b_{(j+1) \cdot \frac{n}{3}-1}}\}$ and $S_i = \bigcup_j S^{i,j}$, where $i, j \in \{0, 1, 2\}$. Let us observe that, analogously to the previous results, if, fixed $i \in \{0, 1, 2\}$, we know the $\frac{n}{3}$ elements of S_i, we can define a Latin subrectangle R_i of L of $\frac{n}{3}$ rows and n columns. Indeed, each R_i is the conveniently ordered (that is, unless principal isotopism) following

$\frac{n}{3} \times n$ array:

$$\begin{pmatrix} l_{a_{i\cdot\frac{n}{3}}b_0} & l_{a_{i\cdot\frac{n}{3}}b_1} & \cdots & l_{a_{i\cdot\frac{n}{3}}b_{n-1}} \\ l_{a_{i\cdot\frac{n}{3}+1}b_0} & l_{a_{i\cdot\frac{n}{3}+1}b_1} & \cdots & l_{a_{i\cdot\frac{n}{3}+1}b_{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ l_{a_{(i+1)\cdot\frac{n}{3}-1}b_0} & l_{a_{(i+1)\cdot\frac{n}{3}-1}b_1} & \cdots & l_{a_{(i+1)\cdot\frac{n}{3}-1}b_{n-1}} \end{pmatrix}$$

Therefore, if we exactly know the elements of S_0, S_1 and S_2, we will obtain L. Indeed, the product of the different ways in which we can fix these three sets is the number of different Latin squares which have Θ as a principal autotopism.

We can start with S_0, which can be fixed of $n!$ different ways. Now, to obtain S_1, we fix in a first step the elements of $S^{1,0}$. This set will contain k elements of $S^{0,1}$ and $\frac{n}{3}-k$ elements of $S^{0,2}$, where k can vary between 0 and $\frac{n}{3}$. That is, we can fix $S^{1,0}$ of $\frac{n}{3}! \cdot \sum_{k=0}^{n/3} \binom{n/3}{k}^2$ ways. Besides, for each of the previous ways, the k elements of $S^{0,2}$ which have not been chosen for $S^{1,0}$ must be in $S^{1,1}$ and the $\frac{n}{3}-k$ elements of $S^{0,1}$ which have not been chosen for $S^{1,0}$ must be in $S^{1,2}$. To complete these sets we must choose k elements of $S^{0,0}$ which will correspond to $S^{1,1}$, corresponding the rest of the elements of $S^{0,0}$ to $S^{1,2}$. So, S_1 can be chosen of $\left(\frac{n}{3}!\right)^3 \cdot \sum_{k=0}^{n/3} \binom{n/3}{k}^3$ different ways.

Finally, to obtain S_2, let us observe that according to the previous process, we know which elements correspond to each $S^{2,j}$ and we only must assign each of them to the corresponding $l_{a_{k\cdot\frac{2n}{3}}b_l}$. So, we can fix S_2 of $\left(\frac{n}{3}!\right)^3$ different ways. Therefore, we finally obtain that:

$$\Delta(\Theta) = n! \cdot \left(\frac{n}{3}!\right)^3 \cdot \sum_{k=0}^{n/3} \binom{n/3}{k}^3 \cdot \left(\frac{n}{3}!\right)^3 = n! \cdot \left(\frac{n}{3}!\right)^6 \cdot \sum_{k=0}^{n/3} \binom{n/3}{k}^3.$$

□

Let us see an example:

Example 2.7. Let us consider $n = 6$ and $N = \{0, 1, 2, 3, 4, 5\}$. There are $15^2 = 225$ principal isotopisms $(\alpha, \beta, \epsilon) \in \mathcal{I}_6$, with α and β being a composition of three cycles of length 2. We will work, for example, with the following principal isotopisms:

$$\Theta_1 = \{((01)(23)(45), (02)(35)(14), \epsilon)\},$$

$$\Theta_2 = \{((02)(14)(35), (01)(23)(45), \epsilon)\}.$$

So:

$$LS(\Theta_1) = \left\{ \begin{pmatrix} a & b & c & d & e & f \\ c & e & a & f & b & d \\ g & h & i & j & k & l \\ i & k & g & l & h & j \\ m & o & p & q & r & s \\ p & r & m & s & o & q \end{pmatrix} \in LS(6) \right\}_{a,b,\ldots,r,s \in N},$$

$$LS(\Theta_2) = \left\{ \begin{pmatrix} a & b & c & d & e & f \\ g & h & i & j & k & l \\ b & a & d & c & f & e \\ m & o & p & q & r & s \\ h & g & j & i & l & k \\ o & m & q & p & s & r \end{pmatrix} \in LS(6) \right\}_{a,b,\ldots,r,s \in N}$$

From Proposition 2.6, $\Delta(\Theta_1) = \Delta(\Theta_2) = 6! \cdot (2!)^6 \cdot \sum_{k=0}^{2} \binom{2}{k}^3 = 460800$. Besides:

$$LS(\Theta_1) \cap LS(\Theta_2) = \left\{ \begin{pmatrix} a & b & c & d & e & f \\ c & e & a & f & b & d \\ b & a & d & c & f & e \\ d & f & b & e & a & c \\ e & c & f & a & d & b \\ f & d & e & b & c & a \end{pmatrix} \in LS(6) \right\}_{a,b,\ldots,k,l \in N},$$

being $|LS(\Theta_1) \cap LS(\Theta_2)| = 6! = 720$. ◁

2.4 Cycles of length $\frac{n}{4}$

Let us now suppose that α and β are both the composition of four cycles of length $\frac{n}{4}$:

$$\alpha = (a_0 a_1 ... a_{\frac{n}{4}-1})(a_{\frac{n}{4}} a_{\frac{n}{4}+1} ... a_{\frac{2n}{4}-1})(a_{\frac{2n}{4}} a_{\frac{2n}{4}+1} ... a_{\frac{3n}{4}-1})(a_{\frac{3n}{4}} a_{\frac{3n}{4}+1} ... a_{n-1}),$$

$$\beta = (b_0 b_1 ... b_{\frac{n}{4}-1})(b_{\frac{n}{4}} b_{\frac{n}{4}+1} ... b_{\frac{2n}{4}-1})(b_{\frac{2n}{4}} b_{\frac{2n}{4}+1} ... b_{\frac{3n}{4}-1})(b_{\frac{3n}{4}} b_{\frac{3n}{4}+1} ... b_{n-1}).$$

To obtain $\Delta(\Theta)$, we will now indicate a possible algorithm to follow. So, to get a Latin square $L \in LS(n)$ which has Θ as a principal autotopism, we can define, fixed $i, j \in \{0,1,2,3\}$ and analogously to the proof of Proposition 2.6, the sets $S^{i,j} = \{l_{a_{i \cdot \frac{n}{4}} b_{j \cdot \frac{n}{4}}}, l_{a_{i \cdot \frac{n}{4}} b_{j \cdot \frac{n}{4}+1}}, ..., l_{a_{i \cdot \frac{n}{4}} b_{(j+1) \cdot \frac{n}{4}-1}}\}$ and $S_i = \bigcup_j S^{i,j}$. Fixed the elements l_{st} corresponding to each S_i, we can obtain a subrectangle R_i of L, in a similar way as we have just done it in the mentioned proof. Therefore, to get L, we must fix all the sets S_i and to do it, we can follow the next algorithm: first, we fix S_0, which can be obtained of $n!$ different ways. Then, we are going to fix the sets S_i with i from 1 to 3, in this order. To obtain each S_i we must fix the sets $S^{i,j}$, with j from 0 to 3, also in this order.

Let us observe that, once we have fixed the elements of $S^{0,j}$ for all $j \in \{0,1,2,3\}$, whenever we want to fix the elements of a set $S^{i,j}$, with $i \neq 0$, we must choose x_t elements of $S^{0,t}$ with $t \in \{0,1,2,3\} \setminus \{j\}$, in such a way that $\sum_t x_t = \frac{n}{4}$. Besides, all these elements must be adequately chosen to obtain finally a Latin square. So, to simplify the notation, we are going to define for each $i \in \{1,2,3\}$ and $j \in \{0,1,2,3\}$:

$$s^{i,j} = \left(s_0^{i,j}, s_1^{i,j}, s_2^{i,j}, s_3^{i,j}\right) \in \left\{0, 1, ..., \frac{n}{4}\right\}^4,$$

such that:

i) $s_j^{i,j} = 0$, for all $i \in \{1,2,3\}$ and $j \in \{0,1,2,3\}$,

ii) $\sum_{t=0}^{3} s_t^{i,j} = \frac{n}{4}$, for all $i \in \{1,2,3\}$ and $j \in \{0,1,2,3\}$,

iii) $\sum_{j=0}^{3} s_t^{i,j} = \frac{n}{4}$, for all $i \in \{1,2,3\}$ and $t \in \{0,1,2,3\}$,

iv) $\sum_{i=1}^{3} s_t^{i,j} = \frac{n}{4}$, for all $j \in \{0,1,2,3\}$ and $t \in \{0,1,2,3\} \setminus \{j\}$.

Then, fixed a subset $A \subseteq S_0$, we will say that we choose $s^{i,j} = (s_0^{i,j}, s_1^{i,j}, s_2^{i,j}, s_3^{i,j})$ elements of $S_0 \setminus A$ to fix the elements which belong to $S^{i,j}$, if we choose $s_0^{i,j}$ ones of $S^{0,0} \setminus A$, $s_1^{i,j}$ ones of $S^{0,1} \setminus A$, $s_2^{i,j}$ ones of $S^{0,2} \setminus A$ and $s_3^{i,j}$ ones of $S^{0,3} \setminus A$. Let us observe that the previous conditions (i) to (iv) are therefore necessary to get a Latin square starting from all the so fixed $S^{i,j}$.

Therefore, the canonical construction path method in this case follows the next algorithm:

Algorithm 2.8.

i) S_0 can be fixed of $n!$ different ways.

ii) To determine $S^{1,0}$ we must choose $(0, s_1^{1,0}, s_2^{1,0}, \frac{n}{4} - s_1^{1,0} - s_2^{1,0})$ elements of S_0, where $s_1^{1,0} + s_2^{1,0} \leq \frac{n}{4}$.

Fixed $s_1^{1,0}$ and $s_2^{1,0}$:

iii) To determine $S^{1,1}$ we must choose $(s_0^{1,1}, 0, s_2^{1,1}, \frac{n}{4} - s_0^{1,1} - s_2^{1,1})$ elements of $S_0 \setminus S^{1,0}$, where $s_2^{1,1} \leq \frac{n}{4} - s_2^{1,0}$ and $\frac{n}{4} - s_1^{1,0} - s_2^{1,0} \leq s_0^{1,1} + s_2^{1,1} \leq \frac{n}{4}$.

iv) To determine $S^{2,0}$ we must choose $(0, s_1^{2,0}, s_2^{2,0}, \frac{n}{4} - s_1^{2,0} - s_2^{2,0})$ elements of $S_0 \setminus S^{1,0}$, where $s_1^{2,0} \leq \frac{n}{4} - s_1^{1,0}$, $s_2^{2,0} \leq \frac{n}{4} - s_2^{1,0}$ and $\frac{n}{4} - s_1^{1,0} - s_2^{1,0} \leq s_1^{2,0} + s_2^{2,0} \leq \frac{n}{4}$.

Fixed $s_0^{1,1}$ and $s_2^{1,1}$:

v) The rest of the $s_1^{1,0} + s_2^{1,0} + s_0^{1,1} + s_2^{1,1} - \frac{n}{4}$ elements of $S^{0,3}$ which we have not yet used to fix $S^{1,0}$ and $S^{1,1}$ must be in $S^{1,2}$. Besides, we must choose $s_0^{1,2}$ elements of $S^{0,0} \setminus S^{1,1}$ and $\frac{n}{2} - s_1^{1,0} - s_2^{1,0} - s_0^{1,1} - s_2^{1,1} - s_0^{1,2}$ elements of $S^{0,1} \setminus S^{1,0}$, where $\frac{n}{4} - s_2^{1,0} - s_0^{1,1} - s_2^{1,1} \leq s_0^{1,2} \leq \frac{n}{4} - s_0^{1,1}$ and $s_1^{1,0} + s_2^{1,0} + s_0^{1,1} + s_2^{1,1} - \frac{n}{4} + s_0^{1,2} \leq \frac{n}{4}$.

Fixed $s_1^{2,0}$ and $s_2^{2,0}$:

vi) To determine $S^{2,1}$ we must choose $(s_0^{2,1}, 0, s_2^{2,1}, \frac{n}{4} - s_0^{2,1} - s_2^{2,1})$ elements of $S_0 \setminus \{S^{1,1} \cup S^{2,0}\}$, where $s_0^{2,1} \leq \frac{n}{4} - s_0^{1,1}$, $s_2^{2,1} \leq \frac{n}{4} - s_2^{1,1} - s_2^{2,0}$ and $\frac{n}{2} - s_1^{2,0} - s_2^{2,0} - s_0^{1,1} - s_2^{1,1} \leq s_0^{2,1} + s_2^{2,1} \leq \frac{n}{4}$. Besides:

a) As to fix $S^{2,1}$ and $S^{1,2}$, we would have used $s_0^{2,1} + s_0^{1,2}$ elements of $S^{0,0}$, to exist $S^{2,2}$ we must also impose that $s_0^{2,1} + s_0^{1,2} \leq \frac{n}{4}$.

b) As to fix $S^{2,0}$ and $S^{1,2}$, we would have used $s_1^{2,0} + \frac{n}{2} - s_1^{1,0} - s_2^{1,0} - s_0^{1,1} - s_2^{1,1} - s_0^{1,2}$ elements of $S^{0,1}$, to exist $S^{2,2}$ we must also impose that $s_1^{2,0} + \frac{n}{2} - s_1^{1,0} - s_2^{1,0} - s_0^{1,1} - s_2^{1,1} - s_0^{1,2} \leq \frac{n}{4}$.

c) As to fix $S^{2,0}$, $S^{2,1}$ and $S^{1,2}$, we would have used $\frac{n}{4} - (s_1^{2,0} + s_2^{2,0} + s_0^{2,1} + s_2^{2,1} - s_1^{1,0} - s_2^{1,0} - s_0^{1,1} - s_2^{1,1})$ elements of $S^{0,3}$, to exist $S^{2,2}$ we must impose that $0 \leq s_1^{2,0} + s_2^{2,0} + s_0^{2,1} + s_2^{2,1} - s_1^{1,0} - s_2^{1,0} - s_0^{1,1} - s_2^{1,1} \leq \frac{n}{4}$.

Finally, fixed $s_0^{2,1}$ and $s_2^{2,1}$:

vii) The rest of the $s_1^{2,0} + s_2^{2,0} + s_0^{2,1} + s_2^{2,1} - s_1^{1,0} - s_2^{1,0} - s_0^{1,1} - s_2^{1,1}$ elements of $S^{0,3}$ which we have not yet used to fix $S^{2,0}$, $S^{2,1}$ and $S^{1,2}$ must be in $S^{2,2}$. Besides, we must choose $s_0^{2,2}$ elements of $S^{0,0} \setminus \{S^{1,2} \cup S^{2,1}\}$ and $\frac{n}{4} - s_1^{2,0} - s_2^{2,0} - s_0^{2,1} - s_2^{2,1} + s_1^{1,0} + s_2^{1,0} + s_0^{1,1} + s_2^{1,1} - s_0^{2,2}$ elements of $S^{0,1} \setminus \{S^{2,0} \cup S^{1,2}\}$, where $\frac{n}{2} - s_2^{2,0} - s_0^{2,1} - s_2^{2,1} - s_0^{1,2} \leq s_0^{2,2} \leq \frac{n}{4} - s_0^{1,2} - s_0^{2,1}$ and $s_1^{2,0} + s_2^{2,0} + s_0^{2,1} + s_2^{2,1} - s_1^{1,0} - s_2^{1,0} - s_0^{1,1} - s_2^{1,1} + s_0^{2,2} \leq \frac{n}{4}$.

After this process, the elements of the sets $S^{1,3}, S^{2,3}, S^{3,0}, S^{3,1}, S^{3,2}$ and $S^{3,3}$ are all determined. So, we can obtain a Latin square L which has Θ as a principal autotopism. To get it, we must only fix in each $S^{i,j}$ the elements which correspond with each $l_{a_{i \cdot \frac{n}{4}} b_{j \cdot \frac{n}{4}+t}}$ with $t \in \{0, 1, ..., n-1\}$. It can be done, once we know which elements are in $S^{i,j}$, of $\frac{n}{4}!$ different ways.

So, if we denote by $\Omega(\Theta)$ the number of different ways in which we can choose the elements that are included in all the subsets $S^{i,j}$ with $i \in \{1,2,3\}$ and $j \in \{0,1,2,3\}$, by following the previous algorithm, we obtain finally the following:

Proposition 2.9. *Let $\Theta = (\alpha, \beta, \epsilon) \in \mathcal{I}_n$, where $n > 4$ is a multiple of 4, be such that α and β are both the composition of four cycles of length $\frac{n}{4}$. So:*

$$\Delta(\Theta) = n! \cdot \left(\frac{n}{4}!\right)^{12} \cdot \Omega(\Theta)$$

□

In the next table, we can see some values of $\Omega(\Theta)$, obtained by computing the previous algorithm with Maple®:

n	8	12	16	20	24	28
$\Omega(\Theta)$	535	60582	10144679	1829667628	362014297870	75689842399097

Let us see an example:

Example 2.10. Let us consider $n = 8$ and $N = \{0, 1, 2, 3, 4, 5, 6, 7\}$ and let us take the following principal isotopisms:

$$\Theta_1 = \{((01)(23)(45)(67), (01)(24)(35)(67), \epsilon)\},$$
$$\Theta_2 = \{((02)(13)(46)(57), (02)(14)(36)(57), \epsilon)\},$$
$$\Theta_3 = \{((04)(15)(26)(37), (03)(15)(26)(47), \epsilon)\}.$$

So:

$$LS(\Theta_1) = \left\{ \begin{pmatrix} a & b & c & d & e & f & g & h \\ b & a & e & f & c & d & h & g \\ i & j & k & l & m & o & p & q \\ j & i & m & o & k & l & q & p \\ r & s & t & u & v & w & x & y \\ s & r & v & w & t & u & y & x \\ z & A & B & C & D & E & F & G \\ A & z & D & E & B & C & G & F \end{pmatrix} \in LS(8) \right\}_{a,b,\ldots,y,z,A,B,\ldots,F,G \in N},$$

$$LS(\Theta_2) = \left\{ \begin{pmatrix} a & b & c & d & e & f & g & h \\ i & j & k & l & m & o & p & q \\ c & e & a & g & b & h & d & f \\ k & m & i & p & j & q & l & o \\ r & s & t & u & v & w & x & y \\ z & A & B & C & D & E & F & G \\ t & v & r & x & s & y & u & w \\ B & D & z & F & A & G & C & E \end{pmatrix} \in LS(8) \right\}_{a,b,\ldots,y,z,A,B,\ldots,F,G \in N},$$

$$LS(\Theta_3) = \left\{ \begin{pmatrix} a & b & c & d & e & f & g & h \\ i & j & k & l & m & o & p & q \\ r & s & t & u & v & w & x & y \\ z & A & B & C & D & E & F & G \\ d & f & g & a & h & b & c & e \\ l & o & p & i & q & j & k & m \\ u & w & x & r & y & s & t & v \\ C & E & F & z & G & A & B & D \end{pmatrix} \in LS(8) \right\}_{a,b,\ldots,y,z,A,B,\ldots,F,G \in N}.$$

From Proposition 2.9, $\Delta(\Theta_1) = \Delta(\Theta_2) = \Delta(\Theta_3) = 8! \cdot (4!)^{12} \cdot 535 = 88355635200$. Besides:

$$LS(\Theta_1) \cap LS(\Theta_2) \cap LS(\Theta_3) = \left\{ \begin{pmatrix} a & b & c & d & e & f & g & h \\ b & a & e & f & c & d & h & g \\ c & e & a & g & b & h & d & f \\ e & c & b & h & a & g & f & d \\ d & f & g & a & h & b & c & e \\ f & d & h & b & g & a & e & c \\ g & h & d & c & f & e & a & b \\ h & g & f & e & d & c & b & a \end{pmatrix} \in LS(8) \right\}_{a,b,c,d,e,f,g,h \in N},$$

being $|LS(\Theta_1) \cap LS(\Theta_2) \cap LS(\Theta_3)| = 8! = 40320$. ◁

2.5 Cycles of length $\frac{n}{k}$

Let us finally study the general case. So, fixed $n \in \mathbb{N}$ and $N = \{0, 1, ..., n-1\}$, let us suppose that α and β are both the composition of k cycles of length $\frac{n}{k}$:

$$\alpha = (a_0 a_1 ... a_{\frac{n}{k}-1})(a_{\frac{n}{k}} a_{\frac{n}{k}+1} ... a_{\frac{2n}{k}-1}) ... (a_{\frac{(k-1)n}{k}} a_{\frac{(k-1)n}{k}+1} ... a_{n-1}),$$

$$\beta = (b_0 b_1 ... b_{\frac{n}{k}-1})(b_{\frac{n}{k}} b_{\frac{n}{k}+1} ... b_{\frac{2n}{k}-1}) ... (b_{\frac{(k-1)n}{k}} b_{\frac{(k-1)n}{k}+1} ... b_{n-1}).$$

To obtain $\Delta(\Theta)$, we can follow a similar algorithm to the previously indicated. So, to get a Latin square $L \in LS(n)$ which has Θ as a principal autotopism, we can define:

$$S^{i,j} = \{l_{a_{i\cdot\frac{n}{k}} b_{j\cdot\frac{n}{k}}}, l_{a_{i\cdot\frac{n}{k}} b_{j\cdot\frac{n}{k}+1}}, ..., l_{a_{i\cdot\frac{n}{k}} b_{(j+1)\cdot\frac{n}{k}-1}}\}; \qquad S_i = \bigcup_{j=0}^{k-1} S^{i,j}, \text{ for all } i, j \in \{0, 1, ..., k-1\}.$$

Then, it is easy to prove the following:

Theorem 2.11. *Fixed $k \in \mathbb{N}$, let $\Theta = (\alpha, \beta, \epsilon) \in \mathcal{I}_n$, where $n > k$ is a multiple of k, be such that α and β are both the composition of k cycles of length $\frac{n}{k}$. Then:*

$$\Delta(\Theta) = n! \cdot \left(\frac{n}{k}!\right)^{k(k-1)} \cdot \Omega(\Theta),$$

where $\Omega(\Theta)$ is 1, if $k = 1$, and the number of different ways in which we can choose the elements that are included in the corresponding subsets $S^{i,j}$, if $k > 1$. □

In the next table we can see the values of $\Omega(\Theta)$ and $\Delta(\Theta)$, if $2 \leq n \leq 9$:

n	k	$\Omega(\Theta)$	$\Delta(\Theta)$	$N(n,n)$
2	1	1	2	2
3	1	6	12	12
4	1	1	24	576
	2	1	96	
5	1	1	120	161280
6	1	1	720	812851200
	2	1	25920	
	3	10	460800	
7	1	1	5040	61479419904000
8	1	1	40320	10877603245908295680
	2	1	23224320	
	4	535	88355635200	
9	1	1	362880	5524751496156892842531225600
	3	56	948109639680	

2.6 Concluding remarks

Although we have studied in this section the case in which Θ is a principal autotopism, an analogous study can be done with the other two possibilities given in Proposition 2.1, that is, $\Theta = (\epsilon, \beta, \gamma)$ or $\Theta = (\alpha, \epsilon, \gamma)$, although in the first one, the canonical construction path must be done with columns blocks in place of row blocks. So, this algorithm and as a consequence, Theorem 2.11, proves indeed that the necessary condition of Proposition 2.1 is also sufficient.

3 Application in Cryptography: $\mathfrak{F}$-critical sets

A *secret sharing scheme* [3], [19] is a method of sharing a secret key K, by giving n pieces of information called *shares* to n participants, in such a way that K can be reconstructed from certain authorized groups of shares and it cannot be done from unauthorized groups of them. The *access structure* Γ is the set of all the previous authorized groups. A *key management scheme* consists of a number of secret sharing schemes, all of them with a common participant, which can have more than one share. In a *multilevel scheme* the participants are ranked in m ranks, in such a way that l_i of them are in the rank r_i for $i \in \{1, ..., m\}$, where $\sum_{i=1}^{m} l_i = n$ and the secret key can be recovered from the shares of the l_i participants of rank r_i.

There are different mathematical models of secret sharing schemes: geometric configurations, polynomial interpolation, block designs, matroids, vector spaces, graphs, etc. One of this model uses critical sets in Latin squares: We fix a Latin square $L = (l_{ij}) \in LS(n)$ which will be the secret key, although its order n is made public. Each share is then a triple $(i, j, l_{ij}) \in L$ and the set of all the used triples is denoted by S. So, if some participants get a critical set of L by sharing its corresponding triples, they will obtain as consequence the secret key L. The access structure is then $\Gamma = \{P \in PLS(n) : P \subseteq \bigcup_S (i, j, l_{ij}) \subseteq L \text{ and } \exists C \in CS(L) \text{ such that } C \subseteq P\}$. In this model all the participants have shares of the same "weight". By the other way, a multilevel scheme can also be analogously given, by placing all the participants in different levels, in such a way that it exists only a critical set in each level. If one participant is in more than one level, then we have an example of a key management scheme.

There are models in which shares are not of the same weight, that is, models in which some shares can offer more information than other ones. It is useful for example in *hierarchical models* in which there exists some need to provide different levels of confidentiality for data. So, in the previous example, we can obtain a hierarchical model if we give to each participant a different number of triples as share. An other possibility would be to consider different types of shares. In this sense, we can study the use of autotopisms of a Latin square as shares of a secret sharing scheme. To do it, let us observe that, as we have seen in previous sections, each autotopism can be associated to a different number of Latin squares. So, the information about L which gives each autotopism is not the same. To give a possible measure of this difference, we give the following:

Definition 3.1. Let $\Theta \in \mathcal{I}_n$. We define the *weight of* Θ *in* $LS(n)$ as:

$$\omega(\Theta) = \begin{cases} 0 \text{ , if } \Delta(\Theta) = 0, \\ \frac{1}{\Delta(\Theta)} \text{ , if } \Delta(\Theta) \neq 0. \end{cases} .$$

By the other way, a set of autotopisms can never define an unique Latin square, because autotopisms are associated to symmetries of Latin squares and so, given a Latin square L associated to a set $\mathfrak{F}$ of autotopisms, every Latin square L' isotopic to L by an isotopism of type (Id, Id, γ) is also associated to $\mathfrak{F}$. So, if we want to define a secret sharing scheme by using autotopisms as shares, we must also use one or more triples of the corresponding

Latin square to finally get the secret key. Indeed, fixed a subgroup $\mathfrak{F}$ of $\mathcal{U}(L)$ it will be necessary to use the triples of a $\mathfrak{F}$-critical set of L. In this sense, it is interesting to extend in a similar way the previous concept of weight to these triples. To do it, as, fixed a triple $T = (i, j, k) \in N^3$, there are $\frac{N(n,n)}{n}$ Latin squares of order n which contain T, it is enough to define the *weight of* T *in* $LS(n)$ as $\omega(T) = \frac{n}{N(n,n)}$.

It can be interesting to extend these concepts to sets of isotopisms and partial Latin squares (as sets of triples), because, in this way, it could be studied the possible relations of interest to cooperate among participants in such a model. Leaving it for a future study, we have therefore interested in the following protocol:

- We fix a Latin square L of order n. The number n is made public, but L is kept secret as the key.
- A set S which is the union of a number of triples and autotopisms of L is defined.
- Each element of S is privately distributed to an unique participant.
- When a group of participants whose shares constitute a subset $\mathfrak{F}$ of $\mathcal{U}(L)$ and a $\mathfrak{F}$-critical set come together, they can reconstruct L and hence, the secret key.

To finish this paper, let us see an example of this protocol:

Example 3.2. Let us consider $L = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 0 & 4 & 5 & 3 \\ 2 & 0 & 1 & 5 & 3 & 4 \\ 3 & 4 & 5 & 0 & 1 & 2 \\ 4 & 5 & 3 & 1 & 2 & 0 \\ 5 & 3 & 4 & 2 & 0 & 1 \end{pmatrix} \in LS(6)$, and the shares:

$$\Theta_1 = ((012)(345), Id, (021)(354)), \qquad \Theta_2 = (Id, (012)(345), (021)(354)),$$

$$\Theta_3 = ((03)(14)(25), (03)(14)(25), Id), \qquad \Theta_4 = (Id, (03)(14)(25), (03)(14)(25)),$$

$$T_1 = (0, 4, 4), \quad T_2 = (1, 1, 2), \quad T_3 = (1, 5, 3), \quad T_4 = (2, 2, 1),$$

$$T_5 = (2, 4, 3), \quad T_6 = (3, 1, 4), \quad T_7 = (3, 2, 5), \quad T_8 = (3, 3, 0),$$

$$T_9 = (4, 0, 4), \quad T_{10} = (5, 3, 2), \quad T_{11} = (5, 5, 1).$$

So, we have that:

$$\omega(\Theta_1) = \omega(\Theta_2) = \frac{1}{25920}, \qquad \omega(\Theta_3) = \omega(\Theta_4) = \frac{1}{460800},$$

$$\omega(T_i) = \frac{6}{812851200} = \frac{1}{135475200}, \text{ for all } i \in \{1, 2, ..., 11\}.$$

We can therefore see that Θ_1 and Θ_2 are the shares which give more information about L. By the other way, there are a lot of possible combinations to reconstruct L, by taking together a subset A of $\mathfrak{F} = \{\Theta_1, \Theta_2, \Theta_3, \Theta_4\}$ and a subset B of $T = \{T_1, T_2, ..., T_{11}\}$. So, for example, if m is the total number of shared shares, we have the following minimal subsets of the corresponding access structure Γ of this secret sharing scheme:

m	A	B	m	A	B
11	–	T	6	$\Theta_1 \cup \Theta_2$	$\{T_1, T_2, T_6, T_8\}$
11	Θ_4	$T \setminus \{T_9\}$	6	$\Theta_1 \cup \Theta_4$	$\{T_2, T_3, T_7, T_9\}$
10	Θ_3	$T \setminus \{T_1, T_{11}\}$	6	$\Theta_2 \cup \Theta_3$	$\{T_3, T_6, T_8, T_{10}\}$
10	$\Theta_3 \cup \Theta_4$	$T \setminus \{T_1, T_9, T_{11}\}$	6	$\Theta_1 \cup \Theta_3 \cup \Theta_4$	$\{T_2, T_4, T_8\}$
9	Θ_1	$T \setminus \{T_5, T_7, T_{10}\}$	5	$\Theta_1 \cup \Theta_2 \cup \Theta_3$	$\{T_1, T_2\}$
9	Θ_2	$T \setminus \{T_1, T_7, T_{10}\}$	5	$\Theta_2 \cup \Theta_3 \cup \Theta_4$	$\{T_1, T_2\}$
7	$\Theta_1 \cup \Theta_3$	$\{T_2, T_3, T_4, T_6, T_9\}$	5	$\Theta_1 \cup \Theta_2 \cup \Theta_4$	$\{T_2, T_4\}$
7	$\Theta_2 \cup \Theta_4$	$\{T_1, T_2, T_4, T_6, T_9\}$	5	$\mathfrak{F}$	$\{T_1\}$

◁

References

[1] Albert, A. A., Quasigroups I, Transactions of the American Mathematical Society 54 (1943) 507 - 519.

[2] S. E. Bammel and J. Rothstein, The number of 9x9 Latin squares, Discrete Math., 11 (1975) 93 - 95.

[3] G. R. Blakley, Safeguarding cryptographic keys. Proc. AFIPS 1979 Natl. Computer Conference, New York, 48, June 1979, pp. 313 - 317.

[4] R. H. Bruck, Some results in the theory of quasigroups, Transactions of the American Mathematical Society 55 (1944) 19-54.

[5] J. A. Cooper, D. Donovan, J. Seberry, Secret Sharing Schemes arising from Latin squares, Bull. Inst. Combin. Appl. 12 (1994) 33 - 43.

[6] D. Curran, G.H.J. van Rees, Critical sets in latin squares, Proceedings of the Eighth Manitoba Conference on Numerical Mathematics and Computing, Winnipeg, Congr. Numer. 22 (1979) 165 - 168.

[7] A. A. Drisko, On the Number of Even and Odd Latin Squares of Order $p + 1$, Advances in Mathematics 128 (1997) 20-35.

[8] R. M. Falcón Ganfornina, Study of Critical Sets in Latin Squares by using the Autotopism Group, submitted (2005).

[9] I. A. Faradzev, Constructive enumeration of combinatorial objects, Problemes Combinatoires des Graphes Colloque International. CNRS 260. CNRS Paris (1978) 131 - 135.

[10] C. Kościelny, Generating quasigroups for cryptographic applications, International Journal of Applied Mathematics and Computer Science 12 (4) (2002) 559 - 569

[11] B.D. McKay, Nauty user's guide (version 1.5), Technical Report TR-CS-90-02, Department of Computer Science, Australian National University, 1990.

[12] B. D. McKay, Isomorph-free exhaustive generation, J. Algorithms 26 (1998) 306 - 324.

[13] B.D. McKay, A. Meynert, W. Myrvold, Small Latin Squares, Quasigroups and Loops, submitted (2004).

[14] B. D. McKay, I. M. Wanless, Latin squares of order eleven. Preprint 2004. http://cs.anu.edu.au/ bdm/papers/ls11.pdf

[15] J. Nelder, Critical sets in Latin squares, CSIRO Division of Math. and Stats, Newsletter 38:4 (1977).

[16] R. C. Read, Every one a winner, Annals Discrete Math. 2 (1978) 107 - 120.

[17] A. Sade, Autotopies des quasigroupes et des systémes associatifs, Arch. Math. 4 No. 1 (1968) 1 - 23.

[18] J. Seberry, Secret sharing and group identification, R & D Studies, Stage 3, Report from the Centre for Computing and Communication Research to Telecom Australia (1990).

[19] A. Shamir, How to share a secret. Comm. ACM 22, No. 11, Nov. 1979, pp. 612 - 613.
[20] M. B. Wells, The number of Latin squares of order 8, J. Combin. Theory 3 (1967) 98 - 99.

Department of Geometry and Topology. University of Seville.
Apdo. 1160. 41080 - Seville, Spain.

Algorithms for the splitting of formal series; applications to alien differential calculus

Frédéric Fauvet Françoise Richard-Jung Jean Thomann

Abstract

We present algorithms which involve both the splitting of formal series solutions to linear ordinary differential equations with polynomial coefficients into a finite sum of subseries which themselves will be solutions of linear ODEs, and the simplification of the recurrence relations satisfied by their coefficients.When coping with series that are solutions of a given differential equation at an irregular – singular point of rank $k \geq 2$, it enables us to reduce the computations to series solutions of an ODE with an irregularity of rank one. In particular, we are able to conduct effective calculations with Écalle's alien derivations for these series. We apply our techniques to some "accelerating functions" of Écalle.

Introduction

The question of decomposing ("splitting") a given formal series which is a solution of some linear differential equation into a finite sum of series which are also solutions of some differential equation is a classical one and is encountered e. g. in formal calculations for dynamical systems or in the quest for obtaining effective estimates for some generating functions. However, even if the theoretical framework for this simple question is well known (Ore extensions, holonomic functions, effective D-modules, etc) the explicit calculations that one might wish to conduct with computer algebra systems are in practice very explosive, even for simple examples. In this paper we describe a pragmatic approach to this problem: we introduce algorithms that imply new procedures, which rely and articulate with existing ones. The calculations have been performed in Maple. These procedures have been first tested on "academic examples", for which they improved significantly upon the existing techniques. Then, we have applied them to series that appear in the complete solution of some linear differential equations with polynomial coefficients which present a particular mathematical interest.

In the neighborhood of an irregular singular point, say at $z \sim \infty$, a linear differential equation with analytic coefficients has a basis of solutions of the form:

$$y(z) = e^{Q(u)} u^{\alpha} f(u) \tag{1}$$

In that expression:

- $u = z^{\frac{1}{\nu}}$ (ν is an integer, $\alpha \in \mathbb{C}$)
- Q is a polynomial in u with a vanishing constant coefficient
- f is a formal series in u^{-1}, with possibly logarithms, more precisely: $f(u) \in \mathbb{C}[[u^{-1}]][\log u]$

The (optimal) ramified variable u, the polynomial Q and the exponent α are formal invariants (in the sense that they depend on the class of the equation modulo a transformation with formal coefficients) of the equation, for which algebraic algorithms have been designed and implemented during the last two decades ([5], [14]). The formal series f, whose coefficients are computable through recurrence relations are generically divergent but the growth of these coefficients is no worse that some Gevrey order ([16]) and resummation techniques are at hand to get approximate solutions of the given equation, with errors that are exponentially small. When the Newton polygon ([16, 15]) of such an equation has just one slope, equal to one, the series can be resummed by applying the Borel–Laplace transforms; when it has one slope equal to k, we have to apply the Borel transform to a function of the new variable z^k. In the case of a multiplicity of slopes the series will be multisummable ([11, 15, 10]) and they can be treated by applying the same mechanism, but with a –finite – succession of stages (there are several *critical times*, in Écalle's language), using convolution operators that involve some special functions : Écalle's "accelerating functions".

In the present paper, we present calculations for situations of a single slope, equal to $k \geq 2$. We are able to compute effectively the action of alien derivations on these series, generalizing the ones made in [8] for equation of rank one (and single level). In rank ≥ 2, such formal–numerical calculations are completely new and in fact, let alone numerical computations such as ours, almost no examples of calculations on resurgent functions of "level k" (meaning divergent series, which are resurgent as functions of some z^k), or worse with a multiplicity of levels, can be found in the litterature (see however [10], [2]). Note, that although we cope with relatively simple examples, results of this sort are not anecdotic: even in the case of linear ODEs with polynomial coefficients, it is only in the last 25 years, thanks to Ramis' works and Écalle's theory of acceleration of resurgent functions that the asymptotics of such solutions have completely been elucidated. The great majority of so called classical functions fall within this class. The paper is organized as follows : in section 1, we introduce very briefly the context that is relevant for the examples we work on : irregular–singular points of linear ODEs, Stokes phenomenon, Ramis' theorem in differential Galois theory, Écalle's alien derivations and accelerating functions. The algorithms for the splitting of series are introduced and described in section 3 and, in section 4, we explicit the calculations we have performed for the accelerating functions C_3 and C_4.

1 Stokes phenomenon, differential Galois groups, accelerating functions

1.1 Stokes phenomenon; alien derivations

The formal series contained in a basis of formal solutions such as in (1) are generically divergent, but they are asymptotic, in some sectors, to some analytic solutions. The comparision of these analytic "resummations" of the series on various sectors gives rise, for a finite number of critical directions, to the Stokes phenomenon ([16]).
An important object attached to such an equation is its (local) Galois differential group, for which we refer to [15]. J. -P. Ramis proved that it is characterized by the "exponential torus", the "formal monodromy" (which are formal invariants that we don't describe here, referring to [11], [16]), and the Stokes matrices. Now, the "important part" in the Galois differential group comes from the Stokes matrices, which are difficult to calculate and involve transcendental constants. These matrices are unipotent, their logarithms are nilpotent matrices that correspond to operators that are derivations acting on the space of series we are working on. These derivations, in turn, can be decomposed into elementary components : the so called alien derivations of Écalle, which can be defined independantly of Galois considerations. Alien derivations are operators, introduced by Jean Écalle ([6]), acting on some spaces of holomorphic functions; they are in fact derivations relatively to a convolution product. A series f solution to an ODE, at an irregular singular point for which the Newton polygon has a single slope, equal to one, will have a convergent Borel transform $\varphi = \hat{B}(f)$ and this germ φ of analytic function can in fact be continued along any broken line from the origin, and we define resurgent functions and alien operators in this context. This property of analytic continuation along any broken line γ starting from the origin of $\mathbb{C}$, going around a finite number of singularities possibly met on γ leads to the general definition of resurgent functions. Let us introduce them in the most simple setting, which is already enough to treat many natural but non trivial examples. The whole point is that our procedures for the splitting of series enable us to decompose a divergent series into a finite sum of series which, as functions of some new variable z^k, are resurgent in the sense explained below.

Definition 1.1. An analytic germ, continuable as above, is called a simple resurgent function if the behaviour of its analytic continuation at any singularity ω is of the form

$$\frac{1}{2i\pi} log(\zeta - \omega) s(\zeta - \omega) + r(\zeta - \omega)$$

where r, s are regular germs.
We denote by $\mathcal{R}$ the space of simple resurgent functions.

Let ω a non zero complex number and d_θ the half line through ω. We are going to define an operator:

$$\Delta_\omega : \mathcal{R} \longrightarrow \mathcal{R}$$

For series belonging to $\mathcal{R}$ and if ω is the only singularity on d_θ, Δ_ω amounts to extracting the singular part s at the singularity ω, up to a factor $2i\pi$ (and thus acts as 0, if there is no singularity at the point ω):

$$\Delta_\omega f(\zeta) = s(\zeta)$$

which is also equal to the *variation* of $2i\pi f$, namely the difference between the continuation by the right and by the left of $2i\pi f$ at the singular point ω. For the general case, the definition of Δ_ω involves an average of the singular parts obtained at ω relatively to the way we go around the singularities that are met between , when performing the analytic continuation the origin and ω.

Alien derivations Δ_ω thus constitute a family of linear operators, indexed by points ω of the complex plane but they are also derivations of the convolutive algebras of resurgent functions and moreover they satisfy a simple commutation relation with the ordinary derivation $\partial = \frac{d}{dz}$, namely: $[\partial, \Delta_\omega] = \omega\, \Delta_\omega$. For all this we refer to [8] and of course to the original papers of Ecalle. We denote by the same symbol the operators that are the pullbacks in the space of formal series $\mathbb{C}[[z^{-1}]]$ or of more general algebras of ramified formal series, which are indispensable in applications, by (the extended)– inverse Borel transform.

For solutions of linear ODEs, the generic case involves only, for each critical time, one singularity as in Definition 1.1 on each singular direction and this will be the situation for the examples described in detail below.

Now, these derivations and also the can be seen as acting on a formal basis of solutions to an equation such as the one above. It acts in the following way, where we denote by $Y(z) = \sum_{i=1}^{n} u_i \exp(\lambda_i z) f_i(z)$ the general solution of the equation, supposed non resonant, and where the $A_{\lambda_i - \lambda_j}$ are constants (see [8] or [6]) :

$$\Delta_{\lambda_i-\lambda_j} f_i = A_{\lambda_i-\lambda_j}\ f_j \quad \text{or, in a compact form:} \quad \Delta_{\lambda_i-\lambda_j} Y = A_{\lambda_i-\lambda_j}\ u_i \frac{\partial}{\partial u_j} Y.$$

There is only a finite number of Δ_ω (for $\omega = \lambda_i - \lambda_j$) that can act non–trivially on Y and that action is expressed as the action of an *ordinary* (meaning non alien!) differential operator in the variables u_i, on the formal integral Y, constituting a simple example of resurgence equation called by Écalle a *bridge equation*, as it throws a bridge between alien and ordinary differential calculus. We thus get a Lie algebra of Galois derivations acting on the vector space of solutions. In fact, they "belong" to the Lie algebra of the differential Galois group of the equation (later on we shall call it the *Lie–Galois algebra*, to be short), and constitute the most important – and most difficult to determine – part of that algebra.

1.2 Accelerating functions

Écalle's accelerating functions can be defined by an integral formula:

$$C_\alpha(t) = \int_\gamma \exp(u - t u^{\frac{1}{\alpha}}) du \quad \text{where } \gamma \text{ is a Hankel contour.}$$

The accelerating functions are used to define convolution operators to sum divergent series with several critical times; they come together with decelerating functions and it was observed by Anne Duval 15 years ago (see [12], in which the results mentionned in the present section can be found. See also [7]) that these (decelerating) functions fall within the class of so-called Faxen integrals and are particular cases of G-functions of Meijer. As such, they can be written in explicit, though complicated, expansions involving the Gamma function. Each C_α is an entire function, with an expansion at the origin :

$$C_\alpha(t) = 2i \sum_{n\geq 0} \sin \frac{n\pi}{\beta} \frac{\Gamma(1+n/\alpha)}{\Gamma(1+n)} t^n \quad \text{with } 1/\alpha + 1/\beta = 1.$$

An accelerating function with a rational α satisfies a simple linear differential equation with polynomial coefficients, namely $C_{q/p}$ is a solution of $A = 0$, where A is the following operator:

$$D^q - (-1)^{q-p} \prod_{j=1}^{p} (\frac{p}{q} tD + j) \quad \text{where } D = \tfrac{d}{dt}.$$

In fact the operator A admits a simple order one left factor, which entails that $C_{q/p}$ belongs to the kernel of the following operator, of order $q-1$:

$$q \prod_{j=1}^{q-1} (\delta - j) - (-1)^{q-p} p t^q \prod_{j=1}^{p-1} (\frac{p}{q}\delta + j) \quad \text{where } \delta \text{ is the Euler operator } t\tfrac{d}{dt}.$$

Such an equation has a single slope at ∞: the formal series solutions at $z \sim \infty$ will be "k –summable", and resurgent with respect to some variable z^k.

The family of accelerating functions $(C_{q/p})$ constitutes an interesting object of study *per se*, and it was already remarked in [12] that it deserves a thorough study. We show below how our algorithms for the splitting of series make possible calculations that pave the way for the determination of the differential Galois groups of the equations above, of low degree.

2 Simplification tools for recurrence equations

2.1 Ore polynomials and series equality

Let $\mathbf{R}$ be a ring and $\sigma : \mathbf{R} \to \mathbf{R}$ be an injective endomorphism of $\mathbf{R}$.
Let δ be a pseudo-derivation w.r.t. σ, that is a map from $\mathbf{R}$ to $\mathbf{R}$ satisfying:

$$\delta(a+b) = \delta a + \delta b, \delta(ab) = \sigma(a)\delta b + \delta a b \quad \text{for any } a, b \in \mathbf{R}.$$

Definition 2.1. The left skew polynomial ring given by σ and δ is the ring $(\mathbf{R}[x], +, .)$ of polynomials in x over $\mathbf{R}$ with the usual polynomial addition, and multiplication given by:

$$xa = \sigma(a)x + \delta a \quad \text{for any } a \in \mathbf{R}.$$

This ring is denoted $\mathbf{R}[x; \sigma, \delta]$ and its elements are called skew polynomials or Ore polynomials.

We refer to [4] for this definition and the first properties of this ring, and to [3] for the arithmetic and algorithmic point of view, when $\mathbf{R}$ is a field (in particular, greatest common right divisor, extended right Euclidean algorithm).

In the following, we will deal with $\mathbf{R} = \mathbb{C}[n]$, τ the automorphism of $\mathbf{R}$ over $\mathbb{C}$ that takes n to $n+1$, and $\mathbf{R}[x;\tau,0]$ the ring of linear ordinary recurrence operators (with polynomial coefficients). In this ring, the multiplication is given by:

$$xa = \tau(a)x, \quad \text{for any } a \in \mathbf{R}.$$

We are interested in formal series $\sum_{n\geq 0} a_n x^n$, such that the coefficients $(a_n)_{n\geq 0}$ are defined by a finite difference equation with polynomial coefficients. That means that the coefficients of the series are defined by:

- $a_0, a_1, \ldots, a_{m-1}$, the first m terms (also called initial conditions), $a_i \in \mathbb{C}$;
- and a recurrence equation

$$P_0(n)a_n + \cdots + P_r(n)a_{n+r} = 0, \forall n \geq m-r,$$

with $P_0, \ldots, P_r \in \mathbb{C}[n]$ and $P_r(\lambda) \neq 0, \forall \lambda \geq m-r \qquad (1)$.

An equivalent manner of writing the condition (1) is to define

$$\begin{aligned} \overline{\lambda} &= \max\{\lambda \in \mathbb{N}, P_r(\lambda) = 0\} \\ &= -1 \quad \text{if } P_r(\lambda) \neq 0, \forall \lambda \in \mathbb{N}, \end{aligned}$$

and to suppose that $m > \overline{\lambda} + r$.
In the following, this series will be represented by the initial conditions $a_0, \ldots, a_{\overline{\lambda}+r}$ and the skew polynomial:

$$x^{\overline{\lambda}+1}(P_0 + P_1 x + \cdots + P_r x^r) = (\tau^{\overline{\lambda}+1}(P_0) + \cdots + \tau^{\overline{\lambda}+1}(P_r)x^r)x^{\overline{\lambda}+1}.$$

Our objective is to *simplify* the skew polynomial defining the previous series: reduce its degree, simplify the coefficients P_i (in particular reduce their degree in n).

2.2 Redundant initial conditions

Proposition 2.2. *Let a be the series defined by $a_0, \ldots, a_{m-1}$ and the skew polynomial $P = (P_0 + P_1 x + \cdots + P_r x^r)x^{m-r}$, with $P_r(\lambda) \neq 0, \forall \lambda \in \mathbb{N}$.*
Suppose that $m > r$, $P_r(-1) \neq 0$ and $P_0(-1)a_{m-r-1} + P_1(-1)a_{m-r} + \cdots + P_r(-1)a_{m-1} = 0$.
Let b be the series defined by $b_0 = a_0, \ldots, b_{m-2} = a_{m-2}, Q = \tau^{-1}(P_0) + \tau^{-1}(P_1)x + \cdots + \tau^{-1}(P_r)x^r)x^{m-r-1}$. Then $a = b$.

2.3 Right factor

Proposition 2.3. *Let a be the series defined by $a_0, \ldots, a_{m-1}$ and a skew polynomial P, of degree m, with $P_m(\lambda) \neq 0, \forall \lambda \in \mathbb{N}$.*
Let b be the series defined by $b_0 = a_0, \ldots, b_{r-1} = a_{r-1}$ and a skew polynomial Q, of degree r.
Suppose that Q is a right factor of P, and that $Q(a) = 0, \forall j \leq m - r - 1$.
Then $a = b$.

Proof. $r < m$; $P = RQ$ with $Q_r(\lambda) \neq 0, \forall \lambda \in \mathbb{N}$ and $\deg(R) = m - r$.

$$\begin{gathered} Q(a)(j) = Q(b)(j), \forall j \leq m - r - 1, \\ R(Q(a)) = RQ(a) = P(a) = 0, \\ \text{and} \qquad R(Q(b)) = R(0) = 0, \\ \text{then} \qquad Q(a) = Q(b). \end{gathered}$$

So $a_0 = b_0, \ldots, a_{r-1} = b_{r-1}$, and $Q(a) = Q(b) = 0$, then $a = b$. □

2.4 Guessing a right factor

Let a be the series defined by $a_0, \ldots, a_{m-1}$ and the skew polynomial $P = P_0 + P_1 x + \cdots + P_m x^m$, with $P_m(\lambda) \neq 0, \forall \lambda \in \mathbb{N}$.
Knowing $a_0, a_1, \ldots, a_{m-1}, a_m, \ldots a_{N-1}$, we determine another polynomial $\overline{Q}$ of degree $\overline{r}$, such that $\overline{Q}(a)(n) = 0, 0 \leq n \leq N - \overline{r} - 1$. We will explain in the next section how to do that.
We consider the polynomials P and $\overline{Q}$ as polynomials with rational coefficients (in $\mathbb{C}(n)[x; \tau, 0]$) and compute a greatest common right divisor $\tilde{Q}$. Suppose that $\tilde{Q}$ is not trivial, and of degree r.
We denote by $\tilde{c}_1$ and $\tilde{c}_2$ two polynomials (in $\mathbb{C}(n)[x; \tau, 0]$) such that $\tilde{c}_1 P + \tilde{c}_2 \overline{Q} = \tilde{Q}$. We put v the lcm of the denominators of the coefficients of $\tilde{c}_1, \tilde{c}_2$ and $\tilde{Q}$. After multiplication by v, we obtain polynomials c_1, c_2 and $\hat{Q}$ in $\mathbb{C}[n][x; \tau, 0]$ such that $c_1 P + c_2 \overline{Q} = \hat{Q}$. So $\hat{Q}(a)(j) = 0, \forall j \leq N - \overline{r} - 1$.
We compute the $\hat{w}$ the gcd of the coefficients of $\hat{Q}$ and divide $\hat{Q}$ by $\hat{w}$ to obtain $Q = Q_0 + Q_1 x + \cdots + Q_r x^r$. Define

$$\begin{aligned} \overline{\lambda} &= \max\{\lambda \in \mathbb{N}, \hat{w} Q_r(\lambda) = 0\} \\ &= -1 \quad \text{if } \hat{w} Q_r(\lambda) \neq 0, \forall \lambda \in \mathbb{N}. \end{aligned}$$

Consider now the series b defined by $b_0 = a_0, \ldots, b_{r+\overline{\lambda}} = a_{r+\overline{\lambda}}$ and the skew polynomial $x^{\overline{\lambda}+1} Q$.
Suppose that $N \geq m - r + \overline{r} + \overline{\lambda} + 1$. Then $a = b$.

Proof. $Q(a)(j) = 0, \forall \overline{\lambda} < j \leq N - \overline{r} - 1$, so $x^{\overline{\lambda}+1}Q(a)(j) = 0, \forall j \leq N - \overline{r} - \overline{\lambda} - 2$. Moreover $x^{\overline{\lambda}+1}Q$ divides (right, in $\mathbb{C}(n)[x;\tau,0]$) $x^{\overline{\lambda}+1}P$, and the coefficients of the quotient have for denominator some shifted of $\tau^{\overline{\lambda}+1}(Q_r)$, so $x^{\overline{\lambda}+1}Q$ is a right factor of $x^{\overline{\lambda}+1}wP$, for a polynomial $w \in \mathbb{C}[n]$ without root in $\mathbb{N}$. We can apply Proposition 2.3. □
Conclusion: we can apply the first subsection "redundant conditions" to the new polynomial $x^{\overline{\lambda}+1}Q$ in order to reduce its degree.

2.5 Guessing a new polynomial Q

At the present time, it is done by two ways:

- solving *by hand* a linear system, for fixed values of r, the degree of $\overline{Q}$ (in x) and of max_d, the maximum of the degrees of the coefficients of $\overline{Q}$ (in n). This is done by the function *diminue_syst* for all values $r \leq \deg(P)$ and $max_d \leq \max_i \deg(P_i)$;
- using the *gfun* package [17] and the function *listtorec*; this is done by the function *diminue_gfun* ; the number of terms N used can be given by the user as parameter, by default it is assigned to a value depending on the degree of P (in x) and the max of the degrees of the coefficients of P in n.

2.6 Examples

1. consider the series $\sum_{n\geq 0} n!(x^{2n} + x^{2n+1})$. As a solution of the differential linear homogeneous equation

   $$(2 + 6x + 3x^2 + x^3)y(x) + (3x^2 + 8x^3 + 3x^4 - 4x - 2)y'(x) + (x^3 + 2x^4 + x^5)y''(x) = 0,$$

   it is defined by DESIR by the 4 first terms $a_0 = a_1 = a_2 = a_3 = 1$ and the skew polynomial

   $$P = (4+j)^2 - 15 - 6j + (-21 - 6j + 2(4+j)^2)x + (-2 - 2j + (4+j)^2)x^2 + (-10 - 4j)x^3 + (-8 - 2j)x^4.$$

   With *diminue_syst*, we find $\overline{Q} = j + j^2 - 2x + (2-2j)x^2 = \tilde{Q} = \hat{Q} = Q$ and $\hat{w} = 1, \overline{\lambda} = 1$. That means that the series can be defined by the first terms $[1, 1, 1, 1]$ and x^2Q.
   With *diminue_gfun* and $N = 20$, we obtain an other polynomial $\overline{Q}$, but which leads to the same Q as before.
   On this example, we didn't reduce the degree of the skew polynomial, but we simplified the coefficients.

2. the Ramis-Sibuya equation is the following differential linear homogeneous equation of order 3:

   $$eq := (3x^3 - 10x^2 - 2x - 4)x^6y'''(x) + (12x^5 - 47x^4 - 16x^3 - 50x^2 - 8x - 8)x^3y''(x)$$
   $$+2(3x^6 - 14x^5 - 12x^4 - 5x^3 - 14x^2 - 6x - 4)xy'(x) + (12x^4 - 14x^3 + 60x^2 + 12x + 8)y(x) = 0.$$

This equation admits a series solution

$$\hat{f}(t) = t(1 + 2t^2 - 7t^3 + 24t^4 + \dots) = \hat{g}(t) + \hat{g}(t^2),$$

where $\hat{g}$ denotes the Euler series $\hat{g}(t) = \sum_{n\geq 0}(-1)^n n! t^n$.

We first consider the series $\frac{\hat{f}}{t}$ (the *regular part* of the series in the internal data furnished by DESIR) defined by the first terms $[1, 0, 2, -7, 24, -118]$ and a skew polynomial of degree 6

$$P = P_0 + P_1 x + P_2 x^2 + P_3 x^3 + P_4 x^4 + P_5 x^5 + P_6 x^6$$

with coefficients P_i of maximum degree 3.

Such a skew polynomial is represented in MAPLE by $OrePoly(P_0, P_1, \dots, P_6)$.
We find a polynomial

$$\overline{Q} = OrePoly\left(\frac{7}{2} + \frac{25}{4}\,j + \frac{13}{4}\,j^2 + \frac{1}{2}\,j^3, \frac{5}{2} + \frac{9}{4}\,j + \frac{1}{2}\,j^2, 7 + \frac{11}{2}\,j + j^2, \frac{5}{2} + j\right),$$

which divides P: $\overline{Q} = \tilde{Q}$. $v = 4$, so that $\hat{Q} = 4\tilde{Q}$, $\hat{w} = 1$ and $Q = \hat{Q}$, $\overline{\lambda} = -1$.
The same series is defined by the first terms $[1, 0, 2]$ and the skew polynomial $Q = 14 + 25j + 13j^2 + 2j^3 + (5 + 2j)(j + 2)x + (28 + 22j + 4j^2)x^2 + (10 + 4j)x^3$.
On this example, we reduce the degree of the polynomial from 6 to 3.

The series $\hat{f}$ is defined by the first terms $[0, 1, 0, 2, -7, 24]$ and the skew polynomial $P_1 = \sum_{i=0}^{6} \tau^{-1}(P_i)x^i)$.
We find a poynomial $\overline{Q}_1$ which is not a factor of P_1, but the right gcd of P_1 and $\overline{Q}_1$ is not trivial. Finally, we find that the series $\hat{f}$ can be defined by the first terms $[0, 1, 0, 2]$ and the polynomial

$$x(j\left(2\,j^2 + 7\,j + 5\right) + (2\,j + 3)\,(1 + j)\,x + (4\,j^2 + 14\,j + 10)x^2 + (4\,j + 6)x^3).$$

So we obtain a polynomial of degree 4.

Remark 2.4. on both these examples, the results are given by *diminue_syst*. The function *diminue_gfun* doesn't give any new polynomial $\overline{Q}$.

3 Splitting of a series

We consider a series $\hat{f}(x) = \sum_{n\geq 0} a_n x^n$, defined by its first terms $a_0, \dots, a_{m-1}$ and the skew polynomial $x^{m-r}(P_0 + P_1 x + \dots + P_r x^r)$, with $P_0 \neq 0$ and $P_r(\lambda) \neq 0, \forall \lambda \in \mathbb{N}$, $\lambda \geq m - r$.
Our goal is to split the series.
Let $\alpha \in \mathbb{N}^*$. Let $q \in \mathbb{N}, 0 \leq q \leq \alpha - 1$ and $\hat{f}_q(x) = \sum_{j\geq 0} a_{j\alpha+q} x^j$, so that

$$\hat{f}(x) = \hat{f}_0(x^\alpha) + \dots + x^{\alpha-1}\hat{f}_{\alpha-1}(x^\alpha).$$

We want to build a recurrence equation with polynomial coefficients satisfied by each of the α subseries $\hat{f}_q$. An algorithmic process to do this has been first introduced by F. Naegele [13] and implemented in $A^\#$.

3.1 Description of the method

The principle is the following: if we suppose that $\hat{f}$ is defined by a recurrence equation of the form

$$Q_0(j)a_j + Q_1(j)a_{j+\alpha} + \cdots + Q_{\overline{r}}(j)a_{j+\overline{r}\alpha} = 0, \qquad (*)$$

then each equation

$$Q_0(j\alpha+q)a_{j\alpha+q}j + Q_1(j\alpha+q)a_{(j+1)\alpha+q} + \cdots + Q_{\overline{r}}(j\alpha+q)a_{(j+\overline{r})\alpha+q} = 0$$

will define the qth subseries.
So it is now sufficient to describe an algorithm that permits to transform the recurrence equation

$$P_0(j)a_j + P_1(j)a_{j+1} + \cdots + P_r(j)a_{j+r} = 0,$$

into a recurrence equation of type $(*)$.
For this, we consider the system of $r(\alpha-1)+1$ linear equations:

$$P_0(j+i)a_{j+i} + P_1(j+i)a_{j+i+1} + \cdots + P_r(j+i)a_{j+i+r} = 0, \text{for } i = 0, \ldots, r(\alpha-1),$$

and we solve it considering as unknown the $r(\alpha-1)$ coefficients a_{j+k} such that α doesn't divide k, that is the terms:

$$a_{j+1}, \ldots, a_{j+\alpha-1}, a_{j+\alpha+1}, \ldots, a_{j+2\alpha-1}, \ldots, a_{j+(r-1)\alpha+1}, \ldots, a_{j+r\alpha-1}.$$

After Gaussian elimination, the system becoms $Ax = b$, with a matrix A in which there is at least one null row for which the corresponding element in b gives the wanted equation.

3.2 Examples

1. we consider the first example of section 2.6, with $\alpha = 2$.
   If the series is defined by the four first terms $[1,1,1,1]$ and the skew polynomial P, the function *scindage* returns two series defined by skew polynomials of degree 4 and maximum degree of the coefficients 8. But after applying *diminue_syst*, we obtain that both subseries are defined by $a_0 = 1$ and the skew polynomial $-1 - j + x$.
   If the same series is defined by the four first terms $[1,1,1,1]$ and the skew polynomial x^2Q, the function *scindage* returns two series defined by skew polynomials of degree 2 and maximum degree of the coefficients 2. Of course, after applying *diminue_syst*, we obtain the same reduced result as before.

2. we consider the second example of section 2.6, with $\alpha = 2$. More precisely, we split in two subseries the series defined by $[1,0,2]$ and the skew polynomial Q. We find that $\hat{f}_0$ is defined by $a_0 = 1$ and the skew polynomial $-2 - 6j - 4j^2 + x$, and that $\hat{f}_1$ is defined by $[0,-7]$ and the skew polynomial

   $$x\left(-16j^4 - 116j^3 - 302j^2 - 334j - 132 - (16j^3 - 96j^2 - 187j - 118)x + (7+4j)x^2\right).$$

3. we can split the series defined by $[0, 1, 0, 2]$ and the skew polynomial Q_1 for big values of α. With $\alpha = 9$ or 10, the result is immediate. For $\alpha = 10$, the recurrence equation for the corresponding first subseries $\hat{f}_0$ is of degree 2 and maximum degree of the coefficient 20. Unfortunately, the function *diminue_syst* takes more time to give no better result.

3.3 From recurrence equation to differential equation

In this section, our goal is to compare our results to the results obtained by M. Barkatou and al. in [1]. In this paper, the authors proposed different methods to find, given a series $\hat{f} = \sum_{n\geq 0} a_n x^n$ solution of an ordinary linear differential equation and given an integer α, a differential equation satisfied by the subseries $\hat{f}^j(x) = \sum_{n\geq 0} a_{j+mr} x^{j+mr}$. So, we treat the same problem, modulo the fact that $\hat{f}^q(x) = x^q \hat{f}_q(x^q)$.
As an example, they take the equation of Ramis-Sibuya, with $\alpha = 2$, and obtain a differential equation of order 5 (and maximum degree of the coefficient 13), satisfied by the two subseries $\hat{f}^0$ and $\hat{f}^1$. If they apply similar methods to non homogeneous linear differential equations, they obtain two (homogeneous) linear differential equations of order 4 (and maximum degree of the coefficients 12) satisfied by each subseries $\hat{f}^0$ and $\hat{f}^1$.
With our method, knowing the initial conditions and a recurrence equation, we use the package *gfun* and the function *diffeqtorec* to perform the Mellin transform and obtain a non homogeneous differential equation satisfied by each subseries. Then an homogeneous differential equation is simply found by differentiating the non homogeneous one divided by the right member.
By this way, we find that the series $\hat{f}_1$ is solution of the following linear equation:

$$-4x^3y'''(x) - 22x^2y''(x) + (1 - 22x)y'(x) - 2y(x) = 0.$$

For the series $\hat{f}_0$, we find the following operator of degree 5:

$$16\,x^7\frac{d}{dx^5}y\,(x) + \left(16\,x^5 + 212\,x^6\right)\frac{d}{dx^4}y\,(x) + \left(80\,x^4 + 762\,x^5\right)\frac{d}{dx^3}y\,(x)$$
$$+ \left(768\,x^4 + 59\,x^3 - 4\,x^2\right)y''\,(x) + \left(9\,x + 132\,x^3\right)y'\,(x) \; - 10\,y\,(x)\,.$$

4 Examples

4.1 Description of the calculations

We are now ready to apply the techniques described above to cope with series S that appear in a basis of formal solutions to some linear differential equation. The calculations that we perform for the reduced equations of order 2 and 3 respectively, of which C_3 and C_4 are solutions are summed up by the following stages:
– We split each series S in subseries.
– For each subseries, we generate a linear ODE with an irregular–regular point with one

critical time, and of rank one at the origin, of which the given subseries is solution.
– Then the Borel transform of this series is also solution of a linear ODE with polynomial coefficients, in *the Borel plane*, by Fourier duality ([8]).
– Finally, we are in a position to apply numerical procedures to estimate the analytic continuation of the subseries, as in [8].

The calculations are detailed and exhaustive for C_3; C_4 we have only room to show the initial stages, but the rest is straighforward, as we have now satisfactory tools for the splitting.

Remark 4.1. We express the equations in the variable x, with $x = 1/z$, in order to use the procedures (Desir, etc) which are designed for a singularity at the origin.

4.2 The accelerating C_3

This function is solution of the following equation:

$$acc3 := 12\,xy'(x) + 3\,x^2y''(x) + 6\,y(x) - \frac{y(x)}{x^3} = 0$$

A basis of formal solutions is given by DESIR under this form:

```
> desir(acc3,y(x),t,10,res);
```

$$[[x(t) = 1/3\,t^2, y(t) = e^{-2\,t^{-3}}\left(1 - \tfrac{5}{144}\,t^3 + \tfrac{385}{41472}\,t^6 - \tfrac{85085}{17915904}\,t^9 + O\left(t^{10}\right)\right)t^{-3/2}]]$$

The variable *ser* contains the regular part of one solution. The function *prepa_scindage* extracts the summability properties and transforms the series into a suitable data structure: it is described by the initial conditions and the polynomial coefficients of the recurrence equation.

```
> ser_prep:=prepa_scindage(ser);
```

the series is 3 summable

$$ser\_prep := [r, [1, 0, 0], [-1/12 + 1/12\ (3/2 + r)^2, 0, 0, 1/3\ (1 + 1/3\,r)\sqrt{3}]]$$

Here is a readable form of the series:

$$1 - \tfrac{5}{48}\sqrt{3}t^3 + \tfrac{385}{1536}\,t^6 - \tfrac{85085}{221184}\sqrt{3}t^9 + O\left(t^{10}\right)$$

The result of the splitting is the following:

```
> S_S := scindage(3,ser_prep);
```

$$S\_S := [[r, [1], [15 + 108\,r + 108\,r^2, 48\sqrt{3} + 48\sqrt{3}r]], [r, [], [1]], [r, [], [1]]]$$

We try to simplify the recurrence equation for the non null series:

```
> s1_s := diminue_syst(S_S[1]);
```

$$s1\_s := [r, [1], [5 + 36\,r + 36\,r^2, 16\sqrt{3} + 16\sqrt{3}r]]$$

A readable form of the split series is as follows:

$$y_1(t) = 1 - \tfrac{5}{48}\sqrt{3}t + \tfrac{385}{1536}\,t^2 - \tfrac{85085}{221184}\sqrt{3}t^3 + \tfrac{37182145}{14155776}\,t^4 + O\left(t^5\right)$$

Differential equation satisfied by the previous series multiplied by t:

$$equa := \left(5\,x - 16\sqrt{3}\right)f(x) + 16\sqrt{3}x\tfrac{d}{dx}f(x) + 36\,x^3\tfrac{d^2}{dx^2}f(x)$$

```
> desir(equa,f(x),t,3);
```

$[[x(t)=t, f_1(t)=t\left(1-\frac{5}{48}\sqrt{3}t+\frac{385}{1536}t^2+O\left(t^3\right)\right)],$
$[x(t)=t, f_2(t)=e^{4/9\frac{\sqrt{3}}{t}}t\left(1+\frac{5}{48}\sqrt{3}t+\frac{385}{1536}t^2+O\left(t^3\right)\right)]]$
For the following, we denote by y_2 the regular part of f_2.
Differential equation satisfied by its Borel transform:

$$equaborel := \left(16\sqrt{3}+72\zeta\right)\frac{d}{d\zeta}\phi(\zeta)+\left(16\sqrt{3}\zeta+36\zeta^2\right)\frac{d^2}{d\zeta^2}\phi(\zeta)+5\phi(\zeta)=0$$

This equation presents two singularities:

$$sing := [0, -4/9\sqrt{3}]$$

A basis of formal solutions at the origin:

```
> frobenius(equaborel,phi(zeta),zeta=0,3);
```

$[\phi_1(\zeta)=1-\frac{5}{48}\sqrt{3}\zeta+\frac{385}{3072}\zeta^2+O\left(\zeta^3\right),$
$\phi_2(\zeta)=\ln(\zeta)\left(1-\frac{5}{48}\sqrt{3}\zeta+\frac{385}{3072}\zeta^2\right)-\frac{13}{24}\sqrt{3}\zeta+\frac{719}{1024}\zeta^2+O\left(\zeta^3\right)]$
The series ϕ_1 is the Borel transform of $ty_1=f_1$.
A basis of solutions in the neighborhood of the non null singularity:

```
> frobenius(equaborel,phi(zeta),zeta=sing[2],3);
```

$[\psi_1(\zeta)=1+\frac{5}{48}\sqrt{3}\zeta+\frac{385}{3072}\zeta^2+O\left(\zeta^3\right),$
$\psi_2(\zeta)=\ln(\zeta)\left(1+\frac{5}{48}\sqrt{3}\zeta+\frac{385}{3072}\zeta^2\right)+\frac{13}{24}\sqrt{3}\zeta+\frac{719}{1024}\zeta^2+O\left(\zeta^3\right)]$
We verify that the series coefficient of $log(\zeta)$ in ψ_2 is $\hat{B}y_2$, the Borel transform of y_2.
By analytic continuation, we perform the connection:

$$\phi_1=\lambda_1\psi_1+\lambda_2\psi_2,\ \text{with}\,\{\lambda_2=-0.1591549446, \lambda_1=0.9241811708\}$$

The *alien derivative* of $\phi_1=\hat{B}f_1$ at the singularity $-4/9\sqrt{3}$ is $2i\pi\lambda_2\hat{B}y_2$.

```
> evalf(2*I*Pi*lambda2);
```

$$-1.000000010\,i$$

This result is coherent with our knowledge about the relationship between the studied differential equation and the Airy equation on one hand, and the exact computation of the Stokes constants for the Airy equation ([11, 9]) on the other hand. Anne Duval noticed 15 years ago the very simple relation between C^3 and the Airy function ([12]). The Galois group of Airy's equation is $SL_2(\mathbb{C})$, one of the rare groups that had been calculated prior to Ramis's theorem (and of N. Katz's results).

4.3 The accelerating C_4

We study now the following equation:

$$acc4 := -36x^2y''(x)-72xy'(x)-4x^3y^{(3)}(x)-24y(x)+\frac{y(x)}{x^4}=0$$

A basis of formal solutions is given by DESIR:

```
> desir(acc4,y(x),t,15,res);
```

$[[x(t)=1/4\,t^3, y(t)=e^{-3t^{-4}}\left(1-\frac{7}{144}t^4+\frac{385}{41472}t^8-\frac{39655}{17915904}t^{12}+O\left(t^{15}\right)\right)t^{-2}]]$

```
> ser_prep:=prepa_scindage(ser);
```

the series is 4 summable

$$ser\_prep := [r, [1, 0, 0, 0, -\tfrac{7}{36}\sqrt[3]{4}, 0, 0, 0], [16\left(-\tfrac{7}{432} - \tfrac{1}{576}r + \tfrac{1}{288}(6+r)^2 - \tfrac{1}{1728}(6+r)^3\right)\sqrt[3]{4}, 0, 0, 0, \tfrac{5}{36} - 1/12\,(6+r)^2, 0, 0, 0, 1/4\,(-2-1/4\,r)\,4^{2/3}]]$$

We perform the splitting:

```
> S_S := scindage(4,ser_prep);
```

$$S\_S := [[r, [1, -\tfrac{7}{36}\sqrt[3]{4}, \tfrac{385}{2592}4^{2/3}], [0, -1760\,2^{2/3} - 2928\,2^{2/3}r - 1536\,2^{2/3}r^2 - 256\,2^{2/3}r^3, -3540 - 2880\,r - 576\,r^2, -648\sqrt[3]{2} - 216\sqrt[3]{2}r]], [r, [], [1]], [r, [], [1]], [r, [], [1]]]$$

In this case, the simplification of the recurrence relation defining the non null series permits to reduce by 1 the degree of the recurrence equation:

```
> diminue_syst(S_S[1]);
```

$$[r, [1, -\tfrac{7}{36}\sqrt[3]{4}], [312\,r + 56 + 384\,r^2 + 128\,r^3, 309\sqrt[3]{2} + 432\sqrt[3]{2}r + 144\sqrt[3]{2}r^2, 108\,2^{2/3} + 54\,2^{2/3}r]]$$

Here is the readable form of the splitted series:

$$1 - \tfrac{7}{36}2^{2/3}t + \tfrac{385}{1296}\sqrt[3]{2}t^2 - \tfrac{39655}{69984}t^3 + \tfrac{665665}{10077696}2^{2/3}t^4 + O\left(t^5\right)$$

Differential equation satisfied by the previous series multiplied by t:

$$equa := \left(28\,2^{2/3}x^3 + 21\,x^2 - 108\sqrt[3]{2}x\right)g'(x) + \left(144\,x^3 + 54\sqrt[3]{2}x^2 + 412\,2^{2/3}x^4\right)g''(x) + \left(144\,x^4 + 384\,2^{2/3}x^5\right)g^{(3)}(x) + 64\,2^{2/3}x^6 g^{(4)}(x) + \left(-21\,x + 108\sqrt[3]{2}\right)g(x) = 0$$

```
> desir(equa,g(x),t,3);
```

$$[g_1(x) = x^2\left(108\sqrt[3]{2} - 309\,x + \tfrac{19825}{36}2^{2/3}x^2 + O\left(x^3\right)\right),$$
$$g_2(x) = x\left(54\sqrt[3]{2} - 21\,x + \tfrac{385}{24}2^{2/3}x^2 + O\left(x^3\right)\right),$$
$$g_3(x) = e^{-3/16\,\frac{\sqrt[3]{2}(-3+i\sqrt{3})}{x}}x\left(1 - \tfrac{42\,i\sqrt{3}x}{-54\sqrt[3]{2}(-3+i\sqrt{3}) - 324\sqrt[3]{2}} + \tfrac{4620\,2^{2/3}x^2}{-31104\sqrt[3]{2} - 7776\sqrt[3]{2}(-3+i\sqrt{3})} + O\left(x^3\right)\right)$$
$$g_4(x) = e^{3/16\,\frac{\sqrt[3]{2}(3+i\sqrt{3})}{x}}x\left(1 + \tfrac{42\,i\sqrt{3}x}{54\sqrt[3]{2}(3+i\sqrt{3}) - 324\sqrt[3]{2}} + \tfrac{4620\,2^{2/3}x^2}{-31104\sqrt[3]{2} + 7776\sqrt[3]{2}(3+i\sqrt{3})} + O\left(x^3\right)\right)]$$

For the following, we denote by y_3 the regular part of g_3.

Differential equation satisfied by the Borel transform of g_1 and g_2:

$$equaborel := \left(108\sqrt[3]{2} + 576\,\zeta + 384\,2^{2/3}\zeta^2\right)\phi''(\zeta) + \left(54\sqrt[3]{2}\zeta + 144\,\zeta^2 + 64\,\zeta^3 2^{2/3}\right)\phi^{(3)}(\zeta) + \left(309 + 412\,2^{2/3}\zeta\right)\phi'(\zeta) + 28\,2^{2/3}\phi(\zeta) = 0$$

This equation presents three singularities:

$$sing := [0, 3/16\sqrt[3]{2}\left(-3 + i\sqrt{3}\right), -3/16\sqrt[3]{2}\left(3 + i\sqrt{3}\right)]$$

A basis of formal solutions at the origin:

$$[\phi_1(\zeta) = \zeta\left(216\sqrt[3]{2} - 309\,\zeta + \tfrac{19825}{108}2^{2/3}\zeta^2 - \tfrac{5551015}{31104}\sqrt[3]{2}\zeta^3 + O\left(\zeta^4\right)\right),$$
$$\phi_2(\zeta) = 54\sqrt[3]{2} - 21\,\zeta + \tfrac{385}{48}2^{2/3}\zeta^2 - \tfrac{39655}{7776}\sqrt[3]{2}\zeta^3 + O\left(\zeta^4\right),$$
$$\phi_3(\zeta) = 2\ln(\zeta)\left(54\sqrt[3]{2} - 21\,\zeta + \tfrac{385}{48}2^{2/3}\zeta^2 - \tfrac{39655}{7776}\sqrt[3]{2}\zeta^3\right) + 216\sqrt[3]{2} - 288\,\zeta + \tfrac{3359}{24}2^{2/3}\zeta^2 - \tfrac{1391917}{11664}\sqrt[3]{2}\zeta^3 + O\left(\zeta^4\right)]$$

The series ϕ_2 is the Borel transform of g_2.
A basis of solutions in the neighborhood of the first non null singularity:

$$[\psi_1(\zeta) = \zeta\left(-6912\,i\sqrt[3]{2}\sqrt{3} - 6912\,\sqrt[3]{2} + \left(9888 - 9888\,i\sqrt{3}\right)\zeta + \tfrac{317200}{27}\,2^{2/3}\zeta^2 + O\left(\zeta^3\right)\right),$$
$$\psi_2(\zeta) = -1728\,\sqrt[3]{2} - 1728\,i\sqrt[3]{2}\sqrt{3} + \left(672 - 672\,i\sqrt{3}\right)\zeta + \tfrac{1540}{3}\,2^{2/3}\zeta^2 + O\left(\zeta^3\right),$$
$$\psi_3(\zeta) = 2\,\ln(\zeta)\left(-1728\,\sqrt[3]{2} - 1728\,i\sqrt[3]{2}\sqrt{3} + \left(672 - 672\,i\sqrt{3}\right)\zeta + \tfrac{1540}{3}\,2^{2/3}\zeta^2\right)$$
$$- 6912\,i\sqrt[3]{2}\sqrt{3} - 6912\,\sqrt[3]{2} + \left(9216 - 9216\,i\sqrt{3}\right)\zeta + \tfrac{26872}{3}\,2^{2/3}\zeta^2 + O\left(\zeta^3\right)]$$

We verify that the series coefficient of $log(\zeta)$ in ψ_3 is *proportional* to $\hat{B}y_3$, the Borel transform of y_3 : an illustration of the bridge equation (1.1).

5 Conclusion

The algorithms for the manipulation and in particular the splitting of formal series that we have described in the present paper have enabled us to perform calculations with alien derivations for series solutions of some linear ODEs of rank $k \geq 1$, with one critical time. Numerical results of this sort are completely new, even in simple situations. The numerical results we give for the accelerating functions C_3 and C_4 yield data that can be used in a straightforward way:
• to perform numerical approximations to the constants of structure of the Lie–Galois algebra of the corresponding equations
or, in another direction,
• to obtain, as in [9], resummation of formal solutions in a "large" sector, past the Stokes lines, as we have a control of the Stokes phenomenon through the coefficient A_ω of the bridge equation.

We stress the fact that the examples that we treat, in the present work and in a continuation of it for other equations of single rank or for cases with several critical times, are neither innocent nor anecdotic : a great number of the so called classical functions are indeed solutions of linear ODEs, and more specifically, the most interesting among them are solutions to equations with one irregular-singular point and at most another singular point which is at worst regular–singular (the class of Hamburger equations; in particular the confluent hypergeometric equations).
In a way, the present computations, although they rely on mathematics that are now considered as well known (Gevrey asymptotics and the like), use in a certain sense the DESIR code up to its limits, and thus yield new results that are consequences of the seminal works made by Jean Della Dora 25 years ago.

1. M.A. Barkatou, F. Chyzak, and M. Loday-Richaud. Remarques algorithmiques liées au rang d'un opérateur différentiel linéaire. In *From Combinatorics to Dynamical Systems*, volume 3 of *IRMA Lectures in Mathematics and Theoretical Physics*, pages 87–129, 2003.

2. Braaksma, B. L. J.; Immink, G. K.; Sibuya, Y. *The Stokes phenomenon in exact asymptotics*, Pacific J. Math. 187 (1999), no. 1, 13–50.

3. M. Bronstein and M. Petrovsek. An introduction to pseudo-linear algebra. *Theoretical Computer Science*, (157):3–33, 1996.

4. P.M. Cohn. *Free Rings and their Relations.* Academic Press, 1971.

5. J. Della Dora, C. Di Crescenzo, and E.Tournier, *An algorithm to obtain formal solutions of a linear homogeneous differential equation at an irregular singular point*, In EUROSAM 82, ed. J. Calmet, volume 144 of Lecture Notes in Computer Science page 273. Springer-Verlag, Berlin and Heidelberg (1982).

6. J. Écalle, *Les fonctions résurgentes*, Vol. 1, 2, 3, Publ. Math. Orsay (1981–85) .

7. J. Écalle, *L'accélération des fonctions résurgentes*, Manuscript, Orsay 1987.

8. F.Fauvet, J. Thomann, *Formal and numerical calculations with resurgent functions*, to appear in Numerical Algorithms, Vol 40, **4**, December 2005, online at http://dx.doi.org/10.1007/s11075-005-5326-5

9. F. Richard–Jung *Représentations graphiques de solutions d'équations différentielles dans le champ complexe*, PhD thesis, Strasbourg 1988.

10. M.Loday-Richaud, *Introduction à la Multisommabilité*, Gazette des Mathématiciens, SMF No.44 (avril 1990).

11. J. Martinet; J.-P. Ramis, *Théorie de Galois différentielle et resommation* in: Computer algebra and differential equations (E. Tournier ed.) Academic Press 1987.

12. J. Martinet; J.-P.Ramis, *Elementary acceleration and multisummability*, Ann. Inst. H. Poincaré Phys. Théor. 54 (1991), no. 4, 331– 401.

13. F. Naegele. *Autour de quelques équations fonctionnelles analytiques.* PhD thesis, Institut National Polytechnique de Grenoble, 1995.

14. F.Jung, F.Naegele, J.Thomann, *An algorithm of multisummation of formal power series, solutions of linear ODE equations* Mathematics and computers in simulation 42 (1996).

15. M. van der Put; M. Singer, *Galois theory of linear differential equations*, Springer 2003.

16. J.-P. Ramis, *Filtration Gevrey sur le groupe de Picard-Vessiot d'une équation différentielle irrégulière*, Informes de Matematica, IMPA, Série A, 045, June 1985, Rio de Janeiro (1985), 1–38.

17. B. Salvy and P. Zimmermann. Gfun: A maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20(2):163–177, june 1994.

Frédéric Fauvet & Jean Thomann
Mathématiques - IRMA
Université Louis Pasteur et CNRS
7, rue Descartes
67084 Strasbourg Cedex (France)
fauvet@math.u-strasbg.fr
thomann@math.u-strasbg.fr

Françoise Richard-Jung
LMC-IMAG, 51 rue des Mathématiques
38041 Grenoble Cedex 9 (France)

Francoise.Jung@imag.fr
http://www-lmc.imag.fr/lmc-cf/Francoise.Jung/

Ramifications and Singularities of Foliations

Pedro Fortuny Ayuso

March 13, 2006

Abstract

We study the behaviour of sequences of blowing-ups under ramifications, the main result being that for any finite sequence π of blowing-ups, there is a ramification morphism ρ such that the elimination of points of indeterminacy $\tilde{\pi}$ of $\pi^{-1} \circ \rho$ is a sequence of blowing-ups *with centers at regular points of the exceptional divisors* and that if π is the reduction of singularities of a holomorphic foliation $\mathcal{F}$, then ρ can be chosen such that $\tilde{\pi}^\star \rho^\star \mathcal{F}$ has only simple singularities. As an application, we give a simplified version of Camacho-Sad's proof of the Separatrix Theorem.

Introduction

Blowing-up techniques have been used successfully since the very beginnings of the study of singularities of algebraic and analytic objects (see [9], although the usage is even older). The most famous example is, obviously, Hironaka's Resolution of Singularities in characteristic zero [14] (and its extension to analytic spaces [3]), together with Zariski's [21] (and Walker's [20]) result for surfaces and for three-dimensional varieties [22], and Abhyankar's positive-characteristic cases (see [2]). One of the main steps in any desingularization process are the *combinatorial sequences* of blowing-ups: those whose centers are intersections of irreducible components of exceptional divisors; they are closely related to toric geometry [13], [17] and *in a vague sense*, they *measure the complexity of the singularity*: in the case of plane curves, they are responsible for the apparition of Puiseux Exponents

In this paper, following the "transgressive" spirit of the conference, we present a different approach to the study of singularities of holomorphic foliations using *ramifications*: maps $\rho : (\mathbb{C}^2, 0) \to (\mathbb{C}^2, 0)$ with local equation $(y_1, y_2) = (x_1^r, x_2)$ for some integer $r \geq 1$. We study the behaviour of sequences of blowing-ups under these transformations, and prove that if $\pi : \mathcal{X} \to (\mathbb{C}^2, 0)$ is a finite sequence of blowing-ups, then there is a ramification ρ such that if $\tilde{\pi} : \mathcal{X} \to (\mathbb{C}^2, 0)$ is the *elimination of points of indeterminacy* of $\pi^{-1} \circ \rho$, then it is a sequence of blowing-ups centered at *regular points* of the exceptional divisor. Thus, *any finite sequence of blowing-ups can be transformed into one without combinatorial subsequences*. This generalizes the fact that any germ of singular curve can be transformed by a ramification into a union of regular components.

When π is the reduction of singularities of a germ of holomorphic foliation $\mathcal{F}$ in $(\mathbb{C}^2, 0)$, then ρ can be chosen such that the pull-back $\tilde{\pi}^\star \rho^\star \mathcal{F}$ has only simple singularities.

As an example, we give in Section 2 a simplified version of Camacho and Sad's proof of the Separatrix Theorem [6, 7, 8, 18], avoiding the (cumbersome) combinatorial arguments.

The technique described in this paper has already been successfully used in contexts related to singularity theory: for studying an oscillation problem related to Thom's Gradient Conjecture (see [12], in which they are called *ramification-rectification morphisms*) and for studying the polar curves of plane holomorphic foliations ([15], [10] and [11]).

1 Reduction of Singularities and Ramifications

Fix a sequence of blowing-ups $\pi == \pi_n \circ \cdots \circ \pi_1 : \mathcal{X} \to (\mathbb{C}^2, 0)$ whose respective exceptional lines are E_i. Given a germ of analytic curve γ at $(\mathbb{C}^2, 0)$, $\pi^\star\gamma$ will denote the *strict transform* of γ by π. The following notation will simplify all the discussions below:

Definition 1.1. We say that π is a *regular tree of blowing-ups* if the center of π_i is a regular points of the exceptional divisor for all i. The sequence π is a *chain of blowing-ups* if the center P_{i+1} of π_{i+1} belongs to E_i. Finally, a regular tree which is a chain will be called a *string of blowing-ups*; its *length* is n.

If π is a regular tree, we say that an irreducible component E_i of the exceptional divisor E is *a son of* component E_j (and that E_j is the father of E_i) if $E_i \cap E_j \neq \emptyset$ and $i > j$.

Definition 1.2. Let $E = \cup_{i=1}^n E_i$ be the exceptional divisor of π. We say that the irreducible component E_j corresponding to π_j is $\pi-$*terminal* if $(E_j \cdot E_j) = -1$ (self-intersection number). In other words, there is no $i > j$ such that the center P_i of π_i belongs to E_j (i.e. E_j has no sons).

The following notation has become common in the present contex:

Definition 1.3. An irreducible analytic curve $\gamma : (\mathbb{C}, 0) \to (\mathbb{C}^2, 0)$ is a *curvette through* E_i if $\pi^\star\gamma$ meets E transversely at E_i (in particular, $\pi^\star\gamma$ is non-singular); notice that *transversely* implies that $\pi^\star\gamma \cap E$ is a non-singular point of E (it is not a crossing of irreducible components of E).

We say that a germ of analytic curve γ is $\pi-$terminal if it is a curvette through a $\pi-$terminal divisor.

Given a regular tree $\pi : \mathcal{X} \to (\mathbb{C}^2, 0)$ and a $\pi-$terminal divisor E, there exists a unique string $\pi_E : \mathcal{X}_E \to (\mathbb{C}^2, 0)$ with terminal divisor $\tilde{E}$ such that there is a commutative diagram

$$\begin{array}{ccc} \mathcal{X} & & \\ \downarrow{\scriptstyle \pi^E} & \searrow^{\pi} & \\ \mathcal{X}_E & \xrightarrow{\pi_E} & (\mathbb{C}^2, 0) \end{array}$$

where π^E is a finite sequence of point blowing-ups and $(\pi^E)^{-1}(\tilde{E}) = E$. This π_E is called a *branch of* π.

The fundamental concept of this paper is that of *ramification*:

Definition 1.4. A *ramification morphism* is a map $\rho : (\mathbb{C}^2, 0) \to (\mathbb{C}^2, 0)$ such that there are local analytic coordinates (u, v) and (x, y) at $(0, 0) \in (\mathbb{C}^2, 0)$ in which

$$(x(u, v), y(u, v)) = \rho(u, v) = (u^r, v)$$

for some positive integer r, called the *ramification order*. These coordinates will be called *adapted to* ρ. The curve $u = 0$ is the *direction of ramification*.

Given a sequence of blowing-ups π and a ramification morphism ρ, we say that ρ is *transversal to* π if there are local coordinate systems (u, v) and (x, y) adapted to ρ such that none of the centers of π is the infinitely near point given by $T_{(0,0)}(x = 0)$. Transversality to a *curve* will mean transversality to its resolution of singularities.

Recall [14, 1] that given a germ of analytic surface $\mathcal{Y}_P$ and $g, h \in \mathcal{O}_{\mathcal{Y}_P}$ relatively prime, with $g(P) = h(P) = 0$, there is a finite sequence of point blowing-ups $\eta : \tilde{\mathcal{Y}} \to \mathcal{Y}$ such that if $f = g/h$, then $f \circ \eta$ is a well-defined holomorphic map (the result in [14] is much more general, but we only need this statement). The minimal η with this property is called the *elimination of points of indeterminacy of* f (see also [19]).

A holomorphic foliation $\mathcal{F}$ in a non-singular analytic surface $\mathcal{X}$ can be identified *locally* with a germ of holomorphic $1-$form ω, whose local expression $a(x, y)dx + b(x, y)dy$ is such that a and b are relatively prime in $\mathbb{C}\{x, y\}$. We will speak (abusing notation) indifferently of $\mathcal{F}$ and ω when there is no possibility of confusion. A *separatrix* of $\mathcal{F}$ is a germ of analytic curve $\gamma : (\mathbb{C}, 0) \to \mathcal{X}$ such that the pull-back $\gamma^\star\omega$ is null. A point P is a *simple singularity* of $\mathcal{F}$ if there are coordinates at P such that $\omega(P) = 0$ and the linear part of the vector field $-b(x, y)\partial/\partial x + a(x, y)\partial/\partial y$ has two different eigenvalues $\mu \neq \lambda \neq 0$ with $\mu/\lambda \notin \mathbb{Q}_{>0}$.

We shall make use of the well-known

Theorem (Seidenberg's reduction of singularities, extended version [16, 5]). *Given a germ of holomorphic foliation $\mathcal{F}$ in $(\mathbb{C}^2, 0)$, there is a finite sequence of blowing-ups $\pi : \mathcal{X} \to (\mathbb{C}^2, 0)$ such that:*

1. *All the singularities of the pull-back $\pi^\star\mathcal{F}$ of $\mathcal{F}$ are simple. [16]*
2. *If $E \subset \pi^{-1}(0, 0)$ is an irreducible component of the exceptional divisor which is* dicritical *(i.e. E is* generically *transversal to $\pi^\star\mathcal{F}$), and $\Gamma \not\subset \pi^{-1}(0, 0)$ is an invariant curve of $\pi^\star\mathcal{F}$ which intersects E, then Γ and E meet transversaly [5].*

Condition (2) can be stated saying that any component of the exceptional divisor which is *generically transversal* to $\mathcal{F}$ is actually *transversal* to $\mathcal{F}$. *A reduction of singularities* of $\mathcal{F}$ is a map π which satisfies (1) and (2). If π is minimal, it will be called *the* reduction of singularities of $\mathcal{F}$. Notice that usually only (1) is required, but we shall impose also (2).

If $\mathcal{F}$ is a germ of foliation in $(\mathbb{C}^2, 0)$, an *$\mathcal{F}-$terminal object* (curvette, curve, divisor...) will be a $\pi-$terminal object for π the reduction of singularities of $\mathcal{F}$. Notice that if $\rho^\star\mathcal{F}$ is the pull-back of $\mathcal{F}$, by ρ, then any separatrix of $\rho^\star\mathcal{F}$ is mapped by ρ into a separatrix of $\mathcal{F}$.

Our main results are the following:

Theorem 1.5. *Let $\pi : \mathcal{X} \to (\mathbb{C}^2, 0)$ be a finite sequence of blowing-ups. There exists a ramification ρ such that if $\tilde{\pi} : \tilde{\mathcal{X}} \to (\mathbb{C}^2, 0)$ is the elimination of points of indeterminacy of $\pi^{-1} \circ \rho : (\mathbb{C}^2, 0) \to \mathcal{X}$, then $\tilde{\pi}$ is a regular tree.*

Moreover, if π is a reduction of singularities of a holomorphic foliation $\mathcal{F}$, then ρ can be chosen such that all the singularities of $\tilde{\pi}^\star \rho^\star \mathcal{F}$ are simple.

And its obvious consequence:

Corollary 1.6. *Given a germ of holomorphic foliation $\mathcal{F}$ in $(\mathbb{C}^2, 0)$, there exits a ramification morphism ρ such that the reduction of singularities of $\rho^\star \mathcal{F}$ is a regular tree.*

Remark 1.7. It is clear that the reduction of singularities of a foliation $\mathcal{F}$ is a regular tree if and only if any $\mathcal{F} - terminal$ curve is regular.

1.1 Elimination of points of indeterminacy of a ramification

Let $\pi : \mathcal{X} \to (\mathbb{C}^2, 0)$ be the blowing up of the origin and $\rho : (\mathbb{C}^2, 0) \to (\mathbb{C}^2, 0)$ any ramification. In suitable systems of coordinates, the equations of the *rational map* $\pi^{-1} \circ \rho$ are

$$(\overline{x}_1, \overline{y}_1) = (u^r, v/u^r) \quad \text{and} \quad (\overline{x}_2, \overline{y}_2) = (u^r/v, v) \tag{1}$$

(in each of the local charts of $\mathcal{X}$). An elementary argument shows that the indeterminacy of $\pi^{-1} \circ \rho$ at the origin is eliminated in exactly r blowing-ups. Let E_1 be the exceptional divisor of $\mathcal{X}$; if $\tilde{\rho} : \tilde{\mathcal{X}} \to \mathcal{X}$ is the elimination of points of indeterminacy of $\pi^{-1} \circ \rho$ and the exceptional divisor of $\tilde{\mathcal{X}}$ is decomposed as $\tilde{E} = E_{11} \cup \cdots \cup E_{1r}$ (in order of apparition of exceptional lines), then there is a point $P \in E_1$ such that $\tilde{\rho}(E_{1i}) = P$ for $i = 1, \ldots, r-1$ whereas $\tilde{\rho}(E_{1r}) = E_1$ and, again in suitable coordinates,

$$(\overline{x}_1, \overline{y}_1) = (u_r^r, v_r) \quad \text{with} \quad E_1 \equiv (\overline{x}_1 = 0) \text{ and } E_{1r} \equiv (u_r = 0) \tag{2}$$

that is, $\tilde{\rho}$ is a ramification of order r, at any $Q \in E_{1r} - E_{1(r-1)}$, in the direction of the exceptional divisor E_{1r}.

Equations (1) and (2) are easily translated to the case of a string π of n blowing ups: one verifies easily that $\tilde{\rho}$ is a string of nr blowing-ups. Let $E = E_1 \cup \cdots \cup E_n$ be the exceptional divisor of π. Applying equation (2) iteratively one sees that the exceptional divisor of $\tilde{\pi}$ can be written $\tilde{E} = (E_{11} \cup \cdots \cup E_{1r}) \cup \cdots \cup (E_{n1} \cup \cdots \cup E_{nr})$ with $\tilde{\rho}(E_{ij}) \subset E_i$. Cover $\mathcal{X}$ and $\tilde{\mathcal{X}}$ with the standard charts:

$$\begin{cases} \mathcal{X} = \left(\overline{U}_1 \cup \cdots \cup \overline{U}_n\right) \cup U_n \\ \tilde{\mathcal{X}} = \left(\overline{U}_{11} \cup \cdots \cup \overline{U}_{1r}\right) \cup \cdots \cup \left(\overline{U}_{n1} \cup \cdots \cup \overline{U}_{nr}\right) \cup U_{nr} \end{cases} \tag{3}$$

where each $\overline{U}_i$ contains (abusing terminology) *the point at infinity* of the exceptional divisor E_i of π_i, and U_n contains the origin of E_n. Also, $\overline{U}_{ij}$ contains the point at infinity of E_{ij} for each i, j and U_{nr} the origin of E_{nr}. There are systems of coordinates (the usual ones) $(\overline{x}_i, \overline{y}_i)$ for $\overline{U}_i$, (x_n, y_n) for U_n, $(\overline{u}_{ij}, \overline{v}_{ij})$ for $\overline{U}_{ij}$ and and (u_{ir}, v_{ir}) for U_{ir} such that:

$$(\overline{u}_{ir}, \overline{v}_{ir}) = \left(\frac{1}{v_{ir}}, u_{ir} v_{ir}\right) \quad \text{and} \quad (\overline{u}_{ij}, \overline{v}_{ij}) = \left(\frac{1}{\overline{v}_{i+1j+1}}, \overline{u}_{i+1j+1} \overline{v}_{i+1j+1}^2\right) \tag{4}$$

and the analog conditions in $\mathcal{X}$. An easy induction proves the following:

Proposition 1.8. *The local expression of $\tilde{\rho}$ in $\overline{U}_{ij}$ and $\overline{U}_i$ is:*

$$(\overline{x}_i, \overline{y}_i) = \tilde{\rho}(\overline{u}_{ij}, \overline{v}_{ij}) = \left(\overline{u}_{ij}^{r-j+1}\overline{v}_{ij}^{r-j}, \overline{u}_{ij}^{j-1}\overline{v}_{ij}^{j}\right). \tag{5}$$

For *chains* of blowing-ups which follow the infinitely near points of a curve with only one Puiseux pair, we have:

Proposition 1.9. *Assume $\pi : \mathcal{X} \to (\mathbb{C}^2, 0)$ is a chain of blowing-ups which is the desingularization of a curve γ with one Puiseux exponent of the form p/q with $(p, q) = \overline{q}$. Then:*

- *There is a ramification ρ transversal to π such that the elimination of points of indeterminacy $\tilde{\rho} : \tilde{\mathcal{X}} \to \mathcal{X}$ of $\pi^{-1} \circ \rho$ is a* string.
- *Moreover, if ρ is of order $r = q\overline{r}$ and Q is the intersection of $\pi^\star\gamma$ with the exceptional divisor $\pi^{-1}(0)$, then $\tilde{\rho}$ is, at $\tilde{\rho}^{-1}(Q)$, a ramification transversal to $\pi^\star\gamma$ of order $\overline{rq}$*

Proof. Write the continuous fraction expansion of $p/q = [a_0, a_1, \ldots, a_h]$. Let π_0 be the maximal string in π starting at the beginning and π^0 the remaining blowing-ups. Consider the diagram

$$\begin{array}{ccc} & & \mathcal{X} \\ & \overset{\tilde{\rho}}{\nearrow} & \downarrow{\scriptstyle \pi^0} \\ \tilde{\mathcal{X}} & \xrightarrow{\tilde{\rho}_0} & \mathcal{X}_0 \\ \downarrow{\scriptstyle \tilde{\pi}_0} & & \downarrow{\scriptstyle \pi_0} \\ (\mathbb{C}^2, 0) & \xrightarrow{\rho} & (\mathbb{C}^2, 0) \end{array} \tag{6}$$

where $\tilde{\mathcal{X}}$ is the elimination of points of indeterminacy of $\pi_0^{-1} \circ \rho$. To be coherent with the notation of Proposition 1.8, assume that π_0 has length n, i.e. $n = a_0 + 1$. We claim that for some r, the map $\tilde{\rho}_0$ lifts holomorphically to $\mathcal{X}$ (in other words, for that r, the *rational* map $\tilde{\rho}$ is actually *holomorphic*). The sequence π^0 starts by blowing-up $E_n \cap E_{n-1}$, so that we need only check that $\tilde{\rho}_0$ *restricted to* $E_{n-1r} \cup E_{n1} \cup \cdots \cup E_{nr-1}$ lifts to $\mathcal{X}$. Consider the matrices

$$A = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix},$$

and, for any $l = 1, \ldots, h-1$ and any $i = a_1 + \cdots + a_l + s$ let $\varphi_i(A, B) = B^s A^{a_l} \cdots B^{a_2} A^{a_1 - 1}$ (assuming l odd, and the corresponding product for l even). The equations of $\tilde{\rho}$ at U_{nj} are

$$\tilde{\rho}(\overline{u}_{nj}, \overline{v}_{nj}) = \left(\overline{u}_{nj}^{e_1^j}\overline{v}_{nj}^{e_2^j}, \overline{u}_{nj}^{f_1^j}\overline{v}_{nj}^{f_2^j}\right) \tag{7}$$

where

$$\begin{pmatrix} e_1^j & e_2^j \\ f_1^j & f_2^j \end{pmatrix} = \varphi_i(A, B) \cdot \begin{pmatrix} r-j+1 & r-j \\ j-1 & j \end{pmatrix}$$

for some $0 \leq i \leq a_1 + \cdots + a_h$ which depends on the point of $\mathcal{X}$ we are looking at. From (7) it is clear that $\tilde{\rho}$ is holomorphic in U_{nj} (that is, has no indetermination point in U_{nj}) if and only if $e_1^j e_2^j \geq 0$ and $f_1^j f_2^j \geq 0$ (i.e. if the exponents in each component have the same sign). Hence, it is enough to find a ramification order r for which all the possible pairs (e_1^j, e_2^j) and (f_1^j, f_2^j) share that property. One proves by induction that for any index i, there are $K_i, L_i, \alpha_i, \beta_i \in \mathbb{Z}$ such that

$$\begin{pmatrix} e_1^j & e_2^j \\ f_1^j & f_2^j \end{pmatrix} = \begin{pmatrix} K_i r + \alpha_i(j-1) & K_i r + \alpha_i j \\ L_i r + \beta_i(j-1) & L_i r + \beta_i j \end{pmatrix}. \tag{8}$$

Let r be a common multiple of $\alpha_i \beta_i$ for all $0 \leq i \leq a_1 + \cdots + a_s$. For this r, the exponents e_1^j, e_2^j are always *consecutive multiples* of α_i so that $e_1^j e_2^j \geq 0$ because either one of them is zero or both have the same sign. The same holds for f_1^j and f_2^j and β_i. Thus, $\tilde{\rho}$ is well-defined for this r, as needed.

The hypothesis on r shows that $\tilde{\rho}^{-1}(Q) \in E_{nj_0}$ for $j_0 + a_0 q = p$. Using the extended Euclidean Algorithm one proves that at this divisor, the corresponding f_j and e_j satisfy $f_2^j = 0$ and $e_2^j = \overline{rq}$ in (7). A simple computation gives now the second statement. □

The proof of the following result is a tedious but straightforward computation:

Lemma 1.10. *Let $\sigma : (\mathbb{C}^2, 0) \to (\mathbb{C}^2, 0)$ be a holomorphic map such that there are local coordinates (x, y) and (u, v) for which $(x, y) = \sigma(u, v) = (u^{e_1} v^{e_2}, u^{f_1} v^{f_2})$ with (e_1, e_2) and (f_1, f_2) as in (8). If $\mathcal{F}$ has a simple singularity at $(0, 0)$, then $\sigma^\star \mathcal{F}$ has a simple singularity at $(0, 0)$.*

We can now give the

Proof of Theorem 1.5. Let $T = F_1 \cup \cdots \cup F_p$ be the union of all the terminal divisors of π, and let $\gamma_1, \ldots, \gamma_p$ be a corresponding family of $\pi-$terminal curves. We may assume that all the γ_i are singular, and that π is given by the sequence $\pi = \pi_p \circ \cdots \circ \pi_1$, where π_i is the blowing-up sequence leading to F_i. It is clearly enough to show that for any i, there is a ramification morphism ρ_i such that $\rho_i^\star \gamma_i$ is a union of regular curves, for $\rho = \rho_p \circ \cdots \circ \rho_1$ would satisfy the thesis. Hence, we need only prove that given an irreducible singular curve γ with desingularization π, there is a ramification ρ such that

1. The curve $\rho^\star \gamma$ is a union of non-singular branches.
2. If $\tilde{\pi} : \tilde{\mathcal{X}} \to (\mathbb{C}^2, 0)$ is the elimination of poins of indeterminacy of $\pi^{-1} \circ \rho$, then it is a regular tree.
3. For the second statement: any $Q \in \tilde{\mathcal{X}}$ whose image $\tilde{\rho}(Q)$ in $\mathcal{X}$ is simple for $\pi^\star \omega$, is simple for $\tilde{\rho}^\star \pi^\star \omega$.

If γ has multiplicity m, then (1) holds for any ramification of order r multiple of m.

Property (2) is a straightforward consequence of propositions 1.8 and 1.9.

Finally, (3), follows from Lemma 1.10, because $\tilde{\rho}$ sends divisors into divisors and separatrices into separatrices and because the reduction of singularities of $\mathcal{F}$ satisfies both properties of the extended version of Seidenberg's theorem. □

2 Application: the Separatrix Theorem

Given a germ of holomorphic foliation $\mathcal{F}$ in $(\mathbb{C}^2,0)$, and a regular separatrix S passing through $Q=(0,0)$, the *Camacho-Sad* index of $\mathcal{F}$ at Q along S is

$$\mathrm{I}_Q(\mathcal{F},S) = -\operatorname{Res}_0\left(\frac{a(x,0)}{b(x,0)}dx\right).$$

(assuming that $\mathcal{F}$ is defined near Q by $\omega = ya(x,y)dx + b(x,y)dy$ and $S\equiv(y=0)$).

The following result is classical (partially known at least since [4]):

Proposition 2.1. *If Q is a simple singularity of $\mathcal{F}$, S is a smooth separatrix of $\mathcal{F}$ through Q with $I_Q(\mathcal{F},S)\neq 0$ then there is another (smooth) separatrix T passing through Q.*

Let now π be the blowing-up of $(0,0)$ and call E to the exceptional divisor. Then

Theorem 2.2 ([6]). *If E is invariant for the pull-back $\pi^\star\mathcal{F}$ of $\mathcal{F}$ and $P_1,\ldots,P_r$ are the singular points of $\pi^\star\mathcal{F}$ in E, then*

$$\sum_{i=1}^{r} I_{P_i}(\mathcal{F},E) = -1 \tag{9}$$

Moreover, if S is invariant for $\mathcal{F}$, $\pi^\star S$ is its pull-back by π and $Q=E\cap S$, then

$$I_Q(\pi^\star\mathcal{F},\pi^\star S) = I_{(0,0)}(\mathcal{F},S) - 1. \tag{10}$$

If S and T are both smooth separatrices of $\mathcal{F}$ and $Q=S\cap T$ is a simple singularity of $\mathcal{F}$, then $\mathrm{I}_Q(\mathcal{F},S) = 1/\mathrm{I}_Q(\mathcal{F},T)$ whenever any of both indices is *non-zero*.

Theorem 2.3 ([6]). *Given a germ of holomorphic foliation $\mathcal{F}$ at $(\mathbb{C}^2,0)$, there is a separatrix for $\mathcal{F}$ passing through $(0,0)$.*

Proof. We assume from now on that the foliation is non-dicritical, as these have not only one but an infinite number of separatrices.

Ramifications send separatrices of $\rho^\star\mathcal{F}$ into separatrices of $\mathcal{F}$ so that, by Corollary 1.6, we need only prove the result for foliations whose reduction of singularities is a regular tree.

Let, thus, $\mathcal{F}$ be such that its reduction of singularities $\pi:\mathcal{X}\to(\mathbb{C}^2,0)$ is a tree and let $E=E_1\cup\cdots\cup E_n$ be the exceptional divisor. By Proposition 2.1, we only need to show that there is a point $Q\in E_i$ for some i with $Q\notin E_j$ for $i\neq j$ such that $\mathrm{I}_Q(\mathcal{F},E_i)\neq 0$. Assume, by contradiction, that there is no such Q. This means that, for any Q in the regular part of E, the corresponding index is 0. Under this assumption, we have:

Remark: If an irreducible component F of E has s sons, then

$$\sum_{Q\in F}\mathrm{I}_Q(\mathcal{F},F) = -s-1, \tag{11}$$

and all the terms in the sum are *rational numbers*. This is (easily) proved by induction on the maximal length of a branch starting from F, using Theorem 2.2 and our assumption

that all the residues in the regular part of F are 0. A direct consequence of this is that if F' is the father of F and $P = F \cap F'$, then

$$\mathrm{I}_P(\mathcal{F}, F') = -1. \tag{12}$$

The first divisor E_1 having no father, all the singularities belonging to it either are the crossing with a son of E_1 or have zero index along E_1. Hence, from the Remark and Equation (12), if $G_1, \ldots, G_r$ are all the sons of E_1, letting $Q_i = E_1 \cap G_i$, we must have

$$-r - 1 \overset{(11)}{=} \sum \mathrm{I}_Q(\mathcal{F}, E_1) = \sum_{i=1}^{r} \mathrm{I}_{Q_i}(\mathcal{F}, E_1) \overset{(12)}{=} \sum_{i=1}^{r} -1 = -r$$

which gives the desired contradiction. □

References

[1] *Complex algebraic surfaces*, London Math. Soc. Student Texts, no. 34, Cambridge University Press, 1996.

[2] S. S. Abhyankar, *Resolution of singularities of arithmetical surfaces*, Arithmetical Algebraic Geometry (Proc. Conf. Purdue Univ., 1963), Harper & Row, New York, 1965, pp. 111–152.

[3] J. M. Aroca, H. Hironaka, and J. L. Vicente, *Desingularization theorems*, Memorias Matemáticas del Instituto Jorge Juan, vol. 30, C.S.I.C., Madrid, 1977.

[4] C. A. Briot and J. C. Bouquet, *Propriétés des fonctions définies par des equations différentielles*, Journal de L'École Polytechnique (1856), no. 36, 133–198.

[5] C. Camacho, A. Lins-Neto, and P. Sad, *Topological invariants and equidesingularization for holomorphic vector fields*, Journal of Differential Geometry **20** (1984), 143–174.

[6] C. Camacho and P. Sad, *Invariant varieties through singularities of holomorphic vector fields*, Ann. of Math. **115** (1982), 579–595.

[7] J. Cano, *An extension of the Newton-Puiseux polygon construction to give solutions of pfaffian forms*, Ann. de L'Institut Fourier (1993), no. 43, 125–142.

[8] ———, *Construction of invariant curves for singular holomorphic vector fields*, Proc. of the AMS **125** (1997), no. 9, 2649–2650.

[9] O. Chisini, *La risoluzione delle singolarità di una superficie mediante transformazioni birazionali dello spazio*, Mem. Accad. Sci. Bologna **VII** (1891), no. 8.

[10] N. Corral, *Courbes polaires d'un feuilletage singulier*, C. R. Acad. Sci. Paris Sér. I Math. **331** (2000), no. 1, 51–54.

[11] N: Corral, *Sur la topologie des courbes polaires de certaines feuilletages singuliers*, Ann. Inst. Fourier (2003).

[12] P. Fortuny and F. Sanz, *Gradient vector fields do not generate twister dynamics*, Journal of Differential Equations **174** (2001), 91–100.

[13] W. Fulton, *Introduction to toric varieties*, Princeton University Press, Princeton, NJ, 1993.

[14] H. Hironaka, *Resolution of singularities of an algebraic variety over a field of characteristic zero*, Ann. of Math. **79** (1964), 109–306.

[15] P. Rouillé, *Sur les polaires des certaines* 1*-formes*, C. R. Acad. Sci. Paris. Série I, Math. **326** (1998), no. 6, 677–680.

[16] A. Seidenberg, *Reduction of singularities of the differential equation* $ady = bdx$, Amer. J. Math. (1968), 248–269.

[17] B. Teissier, *Valuations, deformations and toric geometry*, Resolution of Singularities (A Research Textbook in Tribute to Oscar Zariski) (F. Oort H. Hauser, J. Lipman and A. Quirós, eds.), Progress in Mathematics, vol. 181, Birhkäuser, 2000, Proceedings of the Working Group on Resolution of Singularities, Obergurgl 1997.

[18] M. Toma, *A short proof of a theorem of Camcho and Sad*, Enseign. Math. **45** (1999), 311–316.

[19] Lê Dũng Tràng and C. Weber, *Équisingularité dans les pinceaux de germes de courbes planes et* c^0-*suffisance.*, Enseign. Math. **43** (1997), no. 3-4, 355–380.

[20] R. J. Walker, *Reduction of the Singularities of an Algebraic Surface*, Ann. Math. (1935), no. 36, 336–365.

[21] O. Zariski, *The reduction of singularities of an algebraic surface*, Ann. of Math. **40** (1939), 639–689.

[22] ———, *Reduction of the singularities of algebraic three-dimensional varieties*, Ann. of Math. (2) **45** (1944), 472–542. MR 6,102f

Pedro Fortuny Ayuso.
Colegio Mayor Peñafiel.
Universidad de Valladolid. Spain.
pfortuny@sdf-eu.org
http://pfortuny.sdf-eu.org

Formal power series and polynomial dynamical systems

Mikhail V. Foursov Christiane Hespel

Abstract

In this article we consider the problem of describing the formal power series that arise as generating series of polynomial affine dynamical systems. We introduce new notions of multiset weighted grammars and multiset weighted automata and show that the generating series of polynomial dynamical systems are generated by such grammars. We conjecture moreover that any formal power series generated by a multiset weighted grammar is a generating series of a polynomial dynamical system.

1 Introduction

The notion of formal power series in noncommutative variables was introduced by M.P. Schützenberger [9], in relation to automata and formal languages. Many problems from the theory of formal languages use formal power series : for example, arithmetic problems of the theory of formal languages, study of stochastic processes and of the context-free grammars. The formal power series also represent an interesting tool for solving combinatorial problems : enumeration of planar graphs, permutations and rearrangements in monoids.

Two principal families of formal power series have been studied : rational series and a subfamily of them formed by the recognizable and algebraic series. The rational series were also introduced by M.P. Schützenberger who showed that certain properties of rational series in one variable have a good generalization in noncommutative variables. He established the equivalence between the recognizability and the rationality of proper formal power series.

Another application of formal power series lies in the treatment of dynamical systems. M. Fliess [4] developed the idea that the generating series of a system can be used to code the input/output behavior of the system. This idea, together with the idea that the natural realization of a rational series is a bilinear system, led to the creation the algebraic modeling [7].

The main goal of this article is to describe, similarly to the relationship between the rational series and bilinear systems, the relationship between another class of formal power series and a larger class of affine dynamical systems. In sections 4 and 5, we introduce the multiset weighted grammars and multiset weighted automata. In section 6, we show that the generating series of polynomial dynamical systems are exactly those accepted by the multiset weighted automata. In section 7, we conjecture furthermore that the formal power series generated by multiset weighted grammars are exactly the same as the those accepted by multiset weighted automata.

2 Preliminaries

An *affine dynamical system* is a system of ordinary differential equations of the form

$$\begin{cases} \mathbf{q}'(t) = \mathbf{v}_0(\mathbf{q}) + \sum_{j=1}^{m} \mathbf{v}_j(\mathbf{q}) u_j(t), \\ s(t) = h(\mathbf{q}(t)), \end{cases} \tag{1}$$

where

1. $\mathbf{u}(t) = (u_1(t), \ldots, u_m(t)) \in \mathbb{R}^m$ is the input vector,
2. $\mathbf{q}(t) = (x_1, \ldots, x_n) \in \mathcal{M}$ is the current state, where $\mathcal{M}$ is a real differential manifold,
3. $\{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_m\}$ is a family of smooth vector fields on $\mathcal{M}$,
4. $h : \mathcal{M} \to \mathbb{R}$ is a smooth function called the **observation map**,
5. $s(t) \in \mathbb{R}$ is the output function.

We will be working with the *causal functional* that associates to the set of m input functions (also called commands) $\mathbf{u}(t)$ the corresponding output function $s(t)$. To the commands $u_1(t), u_2(t), \ldots, u_m(t)$ we associate the *alphabet* $\mathcal{Z} = \{z_0, z_1, \ldots, z_m\}$ of $(m+1)$ letters, z_0 being associated to the drift (which we will represent as an additional constant input function $u_0(t) \equiv 1$). To every multi–index $I = (i_1, i_2, \ldots, i_k)$ we associate the word $w = z_I = z_{i_1} z_{i_2} \cdots z_{i_k}$. These words form $\mathcal{Z}^*$, the free monoid over $\mathcal{Z}$. (The empty word is denoted by λ.)

The behavior of causal functionals is uniquely described by two noncommutative power series: the generating series and the Chen series.

The *generating series* $G = \sum_{w \in \mathcal{Z}^*} \langle G | z_I \rangle z_I$ of the system [4] is the geometric contribution and it is independent of the input. Its coefficients $\langle G | z_I \rangle$ are obtained by iteratively applying Lie derivatives corresponding to the vector fields to the observation map and evaluating the resulting expression at the initial state $\mathbf{q}_0$:

$$\langle G | z_I \rangle = \langle G | z_{i_1} z_{i_2} \cdots z_{i_k} \rangle = \mathbf{v}_{i_1} \circ \mathbf{v}_{i_2} \circ \cdots \circ \mathbf{v}_{i_k} \circ h \big|_{\mathbf{q}_0}.$$

(The Lie derivative of the function $f(x_1, \ldots, x_n)$ with respect to the vector field $\mathbf{v} = (v_1, \ldots, v_n)$ is defined by $\mathbf{v}(f) = \sum_i v_i \frac{\partial f}{\partial x_i}$.) The generating series completely describes the causal functional. More precisely, two formal power series define the same functional if and only if they are equal [5, 11].

The *Chen series* $\mathcal{C}_u(t) = \sum_{w \in \mathcal{Z}^*} \langle \mathcal{C}_u(t) | z_I \rangle z_I$ measures the input contribution [2, 3], and is independent of the system. The coefficients of the Chen series are calculated recursively by integration using the following two relations:

- $\langle \mathcal{C}_u(t) | \lambda \rangle = 1$,

- $\langle \mathcal{C}_u(t)|w\rangle = \int_0^t \langle \mathcal{C}_u(\tau)|v\rangle u_j(\tau)d\tau$ for a word $w = vz_j$.

The causal functional $s(t)$ is then obtained locally as the product of the generating series and the Chen series [6]:

$$s(t) = \langle G||\mathcal{C}_u(t)\rangle = \sum_{w\in\mathcal{Z}^*} \langle G|w\rangle\langle \mathcal{C}_u(t)|w\rangle \tag{2}$$

This formula is known as the *Peano–Baker formula*, as well as the *Fliess' fundamental formula*.

3 Rational weighted grammars, rational formal power series and bilinear dynamical systems

A *bilinear dynamical system* is an affine dynamical system (1) such that the coefficients of the vector fields are linear in the states.

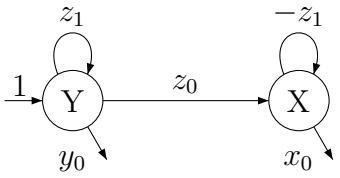
A formal power series $P \in K\langle\langle Z\rangle\rangle$ is called *rational* [1] if it is obtained by a finite numbers of the four main operations (sum, Cauchy product, external product over the ring K and Kleene star) on the individual letters. The rational formal power series are accepted by means of weighted (or multiplicity) automata [10].

The rational series are quite interesting in the study of dynamical system because the generating series of a bilinear dynamical system is rational.

In this section we introduce rational weighted grammars which generate the rational formal power series. Consider for example the following bilinear dynamical system :

$$\begin{cases} y'(t) = x(t) + y(t)u(t), & y(0) = y_0 \\ x'(t) = \quad - x(t)u(t), & x(0) = x_0 \\ s(t) = y(t) \end{cases} \tag{3}$$

Its generating series in accepted by the following weighted automaton :



Definition 3.1. A **rational weighted grammar** over the field **K** is a quintuple (V, Σ, P, S, f), where

- V is a finite set of nonterminal symbols,
- Σ is (another) finite set of terminal symbols,

- S is a distinguished element of V called the start symbol,
- P is a binary relation $V \to K \times \Sigma \times (V \backslash \{S\})$ called the production rules, with the property that the binary relation for the start symbol does not involve any terminal symbols.
- $f : V \to K$ is a function, called the evaluation function.

Remark 3.2. Let us remark that the production rules do not involve any rules of the form $A \to k \in K$, and these rules are grouped into the evaluation function. The reason for this choice will be explain hereafter. The condition on the binary relation is made for convenience, but a grammar that does not satisfy these rules can be easily rewritten in such a form.

Definition 3.3. Let $A \to k_1 z B \in P$, where $k_1 \in K$, $z \in \Sigma$ and $A, B \in V$. Let $w = k_2 \bar{w} A \in K \times \Sigma^* \times V$. Then $k_1 \cdot k_2 \bar{w} z B$ is **derivable** from w, denoted $w \Longrightarrow k_1 \cdot k_2 \bar{w} z B$.

Definition 3.4. The monomial kw (where $k \in K$ and $w \in \Sigma^* \times V$) is said to be **derivable** from S, if it can be obtained by a finite number of derivations starting at S.

Definition 3.5. The **coefficient** k of the word $z \in \Sigma^*$ of the formal power series generated by the grammar is calculated as follows : $k = \sum k_i f(A_i)$, where the sum is over all the words derivable from S of the form $k_i z A_i$, where $k_i \in K$ and $A_i \in V$.

The generating series of the dynamical system (3) is generated by the following grammar :

$$\begin{cases} S \to Z \\ Z \to z_0 X + z_1 Z \\ X \to -z_1 X \\ f(Z) = z_0 \\ f(X) = x_0 \end{cases} \tag{4}$$

Theorem 3.6. *Let*

$$\begin{cases} x_i'(t) = \sum_j a_{ij} x_j(t) + \sum_{j,k} b_{ijk} x_j(t) u_k(t) \\ x_i(0) = x_i \\ s(t) = \sum c_i x_i(t) \end{cases} \tag{5}$$

be a bilinear dynamical system with m inputs $u_1(t), \ldots, u_m(t)$. Then its generating series is generated by the following rational grammar :

$$\begin{cases} S \to c_i X_i \\ X_i \to \sum_j a_{ij} X_j z_0 + \sum_{j,k} b_{ijk} X_j z_k \\ f(X_i) = x_i \end{cases} \tag{6}$$

and recognized by the following weighted automaton :

- *each state is a starting state with the weight* c_i,
- *each state is an output state with the weight* x_i,
- *for the state* X_i *the transitions are* $\delta(X_i, z_0) = \{a_{ij}X_j\}$ *and* $\delta(X_i, z_k) = \{b_{ijk}X_j\}$ *for* $k > 0$.

Proof.

The correspondence between bilinear systems and weighted automata is well-known [8]. The correspondence between weighted automata and rational weighted grammars is obvious from the construction.

□

4 Multiset weighted grammars and multiset formal power series

In this section, we will introduce multiset weighted grammars and explain the reasons why we have chosen this definition.

Definition 4.1. A **multiset** is a grouping of elements with no particular order, in which the elements can appear several times.

Example 4.2. $\{A, A, B\} = \{B, A, A\}$ is a multiset.

Definition 4.3. A **weighted multiset grammar over the field K** is a quintuple (V, Σ, P, S, f), where

- V is a finite set of nonterminal symbols,
- Σ is (another) finite set of terminal symbols,
- S is a distinguished element of V called the start symbol,
- P is a binary relation $V \to K \times \Sigma^* \times \mathbb{M}(V \backslash \{S\})$ called the production rules (where $\mathbb{M}(V \backslash \{S\})$ is the set of multisets over $V \backslash \{S\}$), with the property that the binary relation for the start symbol does not involve any terminal symbols.
- $f : V \to K$ is a function called the evaluation function.

Definition 4.4. Let $A \to k_1 z W \in P$, where $A \in V$, $k_1 \in K$, $z \in \Sigma^*$ and $W \in \mathbb{M}(V)$. Let $w = k_2 \bar{w}(X \cup \{A\} \cup Y) \in K \times \Sigma^* \times \mathbb{M}(V)$, with $k_2 \in K$, $\bar{w} \in \Sigma^*$ and $X, Y \in \mathbb{M}(V)$. Then $w \Longrightarrow k_1 k_2 \bar{w} z(X \cup W \cup Y)$.

In other words, the words are generated from left to right in such a way that the nonterminal symbols are all kept in the multiset.

Definition 4.5. The monomial kw (where $k \in K$ and $w \in \Sigma^* \times \mathbb{M}(V)$) is **derivable** from S, $S \stackrel{*}{\Longrightarrow} kw$, if it can be obtained by a finite number of derivations starting at S.

Definition 4.6. The evaluation function f can be extended to $\tilde{f}$ over $K\langle\Sigma^* \times \mathbb{M}(V)\rangle$ by the following rules:

- $\tilde{f}(A+B) = \tilde{f}(A) + \tilde{f}(B)$, where $A, B \in K\langle\Sigma^* \times \mathbb{M}(V)\rangle$
- $\tilde{f}(kzA) = kz\tilde{f}(A)$ for each $k \in K$, $z \in \Sigma^*$ and $A \in \mathbb{M}(V)$,
- $\tilde{f}(A \cup B) = \tilde{f}(A) \cdot \tilde{f}(B)$, where $A, B \in \mathbb{M}(V)$,
- $\tilde{f}(\{Y\}) = f(Y)$ for each $Y \in V$.

Definition 4.7. The word $w \in \Sigma^*$ belongs to the **support** of the formal power series P generated by the multiset weighted grammar G if there is a derivation $S \stackrel{*}{\Longrightarrow} kwY$ with some $k \in K$ and $Y \in \mathbb{M}(V)$. Usually, $k \neq 0$ is assumed.

Definition 4.8. The coefficient k of the word w belonging to the support of the formal power series P generated by the multiset weighted grammar G is given by $k = \sum k_i \tilde{f}(A_i)$, where the sum is over all the words $k_i w A_i \stackrel{*}{\Longleftarrow} S$, where $k_i \in K$ and $A_i \in \mathbb{M}(V)$.

Definition 4.9. A formal power series generated by a multiset weighted grammar is called a **multiset formal power series**.

Example 4.10. Let us consider the following grammar $\{S \to Y, Y \to zY^2,\ f(Y) = 1\}$.(For simplicity, we will not use the curly brackets to note the multisets.) The derivations proceed as follows :

$$S \Longrightarrow Y \Longrightarrow zY^2 \Longrightarrow z^2Y^3 + zYzY^2 = 2z^2Y^3 \Longrightarrow 6z^3Y^4 \Longrightarrow \cdots$$

The formal power series generated by this grammar is thus

$$L = 1 + z + 2z^2 + 6z^3 + \cdots = \sum_n n! z^n.$$

We have chosen this somewhat unusual definition of grammars for the following reasons (among others), essentially in view of their utilization to describe the generating series of polynomial dynamical systems.

Remark 4.11. The nonrecursiveness of the start symbol and the property that the binary relation for the start symbol does not involve any terminal symbols are not limitations. If these conditions are not satisfied, the grammar can be easily rewritten by adding a new start symbol so that these two properties are satisfied in the new grammar.

Remark 4.12. No class of weighted grammar where all symbols are fully noncommutative seems to correspond to polynomial affine dynamical systems. Consider for example the weighted multiset grammar

$$G_1 = \begin{cases} S \to 2XY, \\ X \to 2z_1Y, \\ Y \to 2z_2X, \\ f(X) = 1, \qquad f(Y) = 1. \end{cases} \tag{7}$$

Now, let us try the two following weighted grammars where all the symbols are fully noncommutative :

$$G_2 = \begin{cases} S \to XY + YX, \\ X \to 2z_1Y, \\ Y \to 2z_2X, \\ f(X) = 1, \qquad f(Y) = 1. \end{cases} \qquad G_3 = \begin{cases} S \to XY + YX, \\ X \to z_1Y + Yz_1, \\ Y \to z_2X + Xz_2, \\ f(X) = 1, \qquad f(Y) = 1. \end{cases}$$

which might seem to be other choices to describe polynomial affine dynamical systems. The languages generated by these three grammars are :

$$\begin{aligned} L_1 &= 2 + 4z_1 + 4z_2 + 16z_1z_2 + 16z_2z_1 + 32z_1z_2z_1 + 32z_2z_1^2 + 32z_1z_2^2 + 32z_2z_1z_2 + \cdots \\ L_2 &= 2 + 4z_1 + 4z_2 + 16z_1z_2 + 16z_2z_1 + 40z_1z_2z_1 + 24z_2z_1^2 + 24z_1z_2^2 + 40z_2z_1z_2 + \cdots \\ L_3 &= 2 + 4z_1 + 4z_2 + 16z_1z_2 + 16z_2z_1 + 32z_1z_2z_1 + 16z_2z_1^2 + 16z_1^2z_2 + 16z_1z_2^2 + \\ &\quad + 16z_2^2z_1 + 32z_2z_1z_2 + \cdots \end{aligned}$$

We have thus $L_1 \neq L_2$ and $L_1 \neq L_3$.

Remark 4.13. It seems to be impossible to devise a grammar that corresponds to a nonlinear dynamical system, without using an evaluation function. Compare for example :

$$G_1 = \begin{cases} S \to Y \\ Y \to zY^2 \\ f(Y) \to 1, \end{cases} \qquad \text{and} \qquad G_2 = \begin{cases} S \to Y \\ Y \to zY^2 + 1 \end{cases}$$

In the second case, the derivations end in the traditional way, i.e. when there are no more nonterminal symbols. The languages generated by these grammars are :

$$\begin{aligned} L_1 &= 1 + z + 2z^2 + 6z^3 + \cdots, \\ L_2 &= 1 + z + 16z^2 + 272z^3 + \cdots \end{aligned}$$

5 Multiset weighted grammar in Greibach normal form and weighted multiset automata

Definition 5.1. A multiset weighted grammar is said to be in **Greibach form** if the rules involve only one letter of the terminal alphabet, that is P is a binary relation $V \to K \times \Sigma \times (V \backslash \{S\})^*$.

We say that such grammars are in Greibach form by analogy to the context–free (ordinary) grammars, even though the languages we consider are not necessarily context–free and even though not all context–free languages can be generated by multiset weighted grammars (for example $\{a^n b^n | n \in \mathbb{N}\}$ cannot be generated by such a grammar).

Definition 5.2. A **weighted multiset automaton** M over the field K is a septuple $(Q, \Sigma, \Gamma, \delta, q_0, F, f)$, where

1. Q is a set of states,
2. Σ is a finite set called the input alphabet,
3. Γ is a finite set called the multiset alphabet,
4. q_0 is the initial state,
5. $F \subseteq Q$ is a set of final states,
6. δ is a transition function from $Q \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\})$ to subsets of $K \times Q \times \mathbb{M}(\Gamma)$, where $\mathbb{M}(\Gamma)$ is the set of multisets over Γ,
7. $f : \Gamma \to K$, is an evaluation function.

The configurations of the automaton are denoted by the quadruple $[q, w, Y, k]$ where q is a state, w the word processed so far, Y the current multiset and k the current monomial coefficient.

The computations in a weighted multiset automaton are effectuated as follows. The multiset is initiated with $\{S\}$ and the monomial coefficient with 1. Now suppose $[k, q_j, B] \in \delta(q_i, a, A)$ and that A is present in the multiset. The above transition causes the automaton to

- change the state from q_i to q_j,
- process (read) the symbol a,
- remove A from the multiset of the automaton,
- join the multiset B to the multiset of the automaton,
- multiply the monomial coefficient by k.

The computation can be terminated at a final state by multiplying the monomial coefficient by $\prod_{i=1}^{n} f(A_i)$, where $\{A_1, \ldots, A_n\}$ is the current state of the multiset of the automaton. If the multiset contains several occurrences of the symbol A, the above rule can be applied to each different occurrence of A and thus it will lead to different derivations.

Definition 5.3. The word $w \in \Sigma^*$ belongs to the **support** of the formal power series P **accepted** by the weighted multiset automaton M if there is a computation $[q_0, \lambda, \{S\}, 1] \overset{*}{\vdash} [q_f, w, Y, k]$ for a final state q_f, a multiset Y and a nonzero coefficient $k \in K$.

Definition 5.4. The coefficient k of the word w from the support of the formal power series P accepted by the weighted multiset automaton M is calculated as the sum of the coefficients of all the possible computations that allow one to process w.

Theorem 5.5. *Let G be the multiset formal power series in Greibach normal form, with the alphabet Σ, accepted by the multiset weighted grammar (V, Σ, P, S, f), where $P = \{S \to k_s W\} \cup \{X \to k_i z_i Y_i\}$. Then it is recognized by the weighted multiset automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, F, f)$, where*

- $Q = \{q_0, q_1, q_2\}$,
- $\Gamma = V$,
- $F = q_2$,
- $\delta(q_0, \lambda, S) = [k_s, q_1, \{W\}]$,
- $[k_i, q_1, \{Y_i\}] \in \delta(q_1, z_i, X)$.
- $[1, q_2, \emptyset] \in \delta(q_1, \lambda, \lambda)$.

Proof.

The proof is immediate from the definitions of weighted multiset grammars in Greibach form and weighted multiset automata.

□

6 Multiset weighted grammars and polynomial affine dynamical systems

Theorem 6.1. *Let*

$$\begin{cases} x_i'(t) = v_{0i}(x_1(t), \ldots, x_n(t)) + \sum_j v_{ji}(x_1(t), \ldots, x_n(t)) u_j(t) \\ s(t) = h(x_1(t), \ldots, x_n(t)), \\ x_i(0) = x_i, \end{cases} \tag{8}$$

be an affine dynamical system with polynomial coefficients v_{ji}. Then its generating series is generated by the following grammar :

$$\begin{cases} S \to h(X_1, \ldots, X_n), \\ X_i \to v_{0i}(X_1, \ldots, X_n) z_0 + \sum_j v_{ji}(X_1, \ldots, X_n) z_j, \\ f(X_i) = x_i, \end{cases} \tag{9}$$

where $V = \{X_1, \ldots, X_n\}$.

Proof.

The proof is done by induction. The coefficient of the empty word is the same using both methods.

Now suppose that for each monomial $z_I = z_{i_1} \cdots z_{i_k}$ of length $k \leq m$, we have $\mathbf{v}_{i_k} \circ \mathbf{v}_{i_{k-1}} \circ \cdots \circ \mathbf{v}_{i_1} \circ h = \sum_j c_j \prod_p x_p^{l_{jp}}(t)$ and $S \stackrel{*}{\Longrightarrow} \sum_j c_j \prod_p X_p^{l_{jp}} z_I$, where all the possible derivation are effectuated in the grammar.

Take $z_J = z_I z_s$ for a $z_s \in \Sigma$. The coefficient of z_J in the generating series is

$$\mathbf{v}_s\Big(\sum_j c_j \prod_p x_p^{l_{jp}}(t)\Big) = \sum_j c_j \sum_q l_{jq} v_{sq}(x_1(t), \ldots, x_n(t)) \prod_p x_p^{l_{jp}-\delta_{pq}}(t) \tag{10}$$

On the other hand, using all the possible derivations involving z_s we obtain :

$$\sum_j c_j \prod_p X_p^{l_{jp}} z_I \Longrightarrow \sum_j c_j \sum_q l_{jq} v_{sq}(X_1, \ldots, X_n) \prod_p X_p^{l_{jp}-\delta_{pq}} z_I z_s. \tag{11}$$

Applying the evaluation function to (11) and evaluating (10) at the initial conditions, we see that the coefficient of the generating series of (8) is the same as the coefficient of the series generated by the multiset weighted grammar (9).

□

Let us remark that many nonpolynomial dynamical systems can be rewritten as polynomial ones. Suppose, for example, that the dynamical system involves a function $f(x(t))$ which is a solution of a polynomial system of autonomous ODEs. Then we can introduce a new variable $\tilde{x}(t)$ and add an additional equation that is satisfied by $f(x)$. For example, $e^{x(t)}$ can be replaced by $\tilde{x}(t)$ inside the dynamical system, together with an additional equation $\tilde{x}'(t) = \tilde{x}(t)$.

Of course, not all affine dynamical systems can be rewritten in polynomial form. Consider, for example, the function $f(t) = \sum_n a_n t^n$, where $a_n = 1$ for prime n and 0 otherwise. The generating series of the dynamical system $y'(t) = f(y(t))$ cannot be generated by a finite weighted grammar (otherwise, we would get an easy way to calculate algorithmically all prime numbers !)

7 Reduction of multiset weighted grammars to Greibach normal form

Conjecture 7.1. *A multiset weighted grammar can be reduced to a Greibach normal form (but it is not necessarily unique).*

We are unable so far to prove this conjecture, but, at least in the case of one–letter grammars, once the generated formal power series is identified, it is often not too hard to find (using a computer algebra software) an affine dynamical system whose generating series is exactly this series.

Example 7.2. Take the grammar $G = \{S \to Y, Y \to z^2Y^2, f(Y) = y_0\}$. It generates the formal power series $R = \sum_n n! y_0^{n+1} z^{2n}$. The Peano-Baker formula gives us immediately that

$$s(t) = y_0 + \frac{1}{2} y_0 \sqrt{\pi y_0}\ t\ exp\left(\frac{y_0 t^2}{4}\right) erf\left(\frac{\sqrt{y_0}t}{2}\right), \tag{12}$$

where $erf'(t) = 2e^{-t^2}/\sqrt{\pi}$.

$s(t)$ satisfies the following differential equation:

$$s''(t) = \frac{(ts'(t) - s(t))(ts'(t) + 2s(t))}{t^2 s(t) - 2}, \qquad s(0) = y_0, s'(0) = 0, \tag{13}$$

which is not written as a dynamical system. However it can be easily rewritten as :

$$\begin{cases} y'(t) = x(t)(t^2 y(t) - 2), & y(0) = y_0, \\ x'(t) = -\dfrac{2y^2(t)}{(t^2 y(t) - 2)^2} - \dfrac{t x(t) y(t)}{t^2 y(t) - 2}, & x(0) = 0, \end{cases} \tag{14}$$

whose generating series is generated by the grammar in Greibach form:

$$\begin{cases} S \to Y \\ Y \to zX(T^2 Y - 2) \\ X \to -2zY^2 D^2 - zTXYD \\ T \to z \\ D \to -z(2TY + T^2 X(T^2 Y - 2))D^2 \\ f(Y) = y_0, \quad f(X) = 0, \quad f(T) = 0, \quad f(D) = -1/2, \end{cases} \tag{15}$$

where T represents t and D represents $1/(t^2 y(t) - 2)$.

This Greibach grammar is not unique, another one can be obtained from the differential equation:

$$\begin{cases} y'(t) = x(t), & y(0) = y_0, \\ x'(t) = \dfrac{(tx(t)y(t) - y(t))(tx(t) + 2y(t))}{t^2 y(t) - 2}, & x(0) = 0, \end{cases} \tag{16}$$

whose solution is also $y(t)$. Other ODEs are also possible.

Remark furthermore that the following grammars in Greibach form, which might seem to be equivalent to G, generate in fact different formal power series :

$$\begin{cases} Y \to zZ, \\ Z \to zY^2, \end{cases} \qquad \begin{cases} Y \to zYZ, \\ Z \to zY, \end{cases} \qquad \begin{cases} Y \to zZY^2, \\ Z \to z. \end{cases} \tag{17}$$

The easiest way to check this is to write corresponding dynamical systems.

8 Conclusions

In this article we have presented a new type of weighted grammars and weighted automata generating (respectively accepting) the formal power series which are generating series of polynomial affine dynamical systems. We conjecture moreover that a more general type of weighted grammars generate the same formal power series.

An interesting direction to follow is to find a criterion allowing one to verify whether there exists a weighted multiset automaton accepting a given formal power series. Another direction could be to describe all the formal power series that arise as generating series of an affine dynamical system, not necessarily polynomial.

Acknowledgments

The authors would like to thank Didier Caucal and Arnaud Carayol for helpful discussions.

References

[1] J. Berstel and C. Reutenauer, *Les séries rationnelles et leurs langages*, Masson, 1984.

[2] K.-T. Chen, *Algebras of iterated path integrals and fundamental groups*, Trans. Am. Math. Soc. **156** (1971), 359–379.

[3] K.-T. Chen, *Iterated path integrals*, Bull. Am. Math. Soc. **83** (1977), 831–879.

[4] M. Fliess, *Fonctionnelles causales non linéaires et indéterminées non commutatives*, Bull. Soc. Math. France **109** (1981), 3–40.

[5] M. Fliess, *On the concept of derivatives and Taylor expansions for nonlinear input/output systems*, in "IEEE Conference on Decision and Control" (San Antonio, Texas), 1983, 643–648.

[6] M. Fliess, *Réalisation locale des systèmes non linéaires, algèbres de Lie filtrées transitives et séries génératrices*, Invent. Math. **71** (1983), 521–537.

[7] C. Hespel, *Une étude des séries formelles non commutatives pour l'approximation et l'identification de systèmes dynamiques*, HDR thesis, Université de Lille–I, 1998.

[8] G. Jacob, *Réalisation des systèmes réguliers (ou bilinéaires) et séries génératrices non commutatives*, Séminaire d'Aussois, RCP567, Outils et modèles mathématiques pour l'automatique, l'analyse de systèmes et le traitement du signal (CNRS Landau), 1980.

[9] M.P. Schützenberger, *Un problème de la théorie des automates*, séminaire Dubreil-Pisot **3** (1959–1960).

[10] M.P. Schützenberger, *On the definition of a family of automata*, Inform.Contr. **4** (1961), 245–270.

[11] Y. Wang and E.D. Sontag, *On two definitions of observation spaces*, Syst. Contr. Lett. **13** (1988), 279–289.

[12] Y. Wang and E.D. Sontag, *Algebraic differential equations and rational control systems*, SIAM J. Contr. Optim. **30** (1992), 126–1149.

[13] Y. Wang and E.D. Sontag, *Generating series and nonlinear systems; analytic aspects, local realizability, and I/O representations*, Forum Math. **4** (1992), 299–322.

Mikhail V. Foursov
IRISA/Université de Rennes–1
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
foursov@irisa.fr

Christiane Hespel
IRISA/INSA de Rennes
20, avenue des Buttes de Coësmes
35043 Rennes Cedex, France
hespel@irisa.fr

Noncommutative computing and rational approximation of multivariate series

Christiane Hespel Cyrille Martig

Abstract

For any multivariate series $s \in \boldsymbol{R}[[X_i]]_{1\leq i\leq n}$, we propose an algorithm for computing a family of rational series $(s_k)_{k\in\boldsymbol{N}}$ such that the difference $s - s_k$ is at least of order k. The method consists in using the computation in noncommutative variables: for any order k, we construct a rational (recognizable) series $s_{nc_k} \in \boldsymbol{R}\langle\langle X_i\rangle\rangle_{1\leq i\leq n}$ in noncommutative variables, of minimal rank r_k, such that s_k is its commutative image. With this object, we provide the Hankel matrix $H(s_{nc_k})$ of the series s_{nc_k} by computing the coefficients of every word $w \in \{X_i\}^*_{1\leq i\leq n}$ from the coefficients of s, in order to maintain the linear dependence relations existing in the already identified part of the Hankel matrix $H(s_{nc_k})$. And then, we construct a noncommutative rational series s_{nc_k} of minimal rank, such that its commutative image s_k is equal to s, up to order k (for every monomial of total degree $\leq k$). By computing the commutative image of the regular expression associated with s_{nc_k}, we rewrite s_k as a quotient of 2 polynomials. The rational series s_k is then provided as a quotient of 2 polynomials.
If s is rational, there is an order k_0 such that $\forall k \geq k_0$, the rank of s_{nc_k} is r_{k_0}. And then for $k \geq k_0$, the commutative image of s_{nc_k} is s, provided as a quotient of 2 polynomials.

Introduction

The starting point is the following : Marie-Françoise Roy has set us the problem of writing a rational multivariate series in the form of a polynomial quotient. This is a subject of interest particularly for handling multivariate series in the Symbolic Computation.
We chose to use the noncommutative computation, the Schützenberger theorem allowing us to reduce a rational series to a recognizable series. By taking this property into account, the noncommutative series seemed the best computing tool. When dealing with this problem by this method, we had to construct some rational approximants of multivariate series : the approximant of the multivariate series s up to order k is described as a pair of polynomials (P_k, Q_k), the coefficients being in $\boldsymbol{R}$ (and not in a semiring [4]). For a given k, this pair is not generally unique. Moreover, this approximant is not a multivariate Padé approximant [2] computed by a gaussian elimination or by Gröbner bases [5], is not a generalization of the Padé approximant [3], but a rational approximant s_k obtained by computing its noncommutative associated series s_{nc_k} of minimal rank. So, when this rank is small, the method is all the more efficient because the basis vectors are indexed by short words.

When s is rational, the family $(P_k/Q_k)_{k\in\boldsymbol{N}}$ converges towards s, is stationary and there is an order k_0 such that $\forall k \geq k_0$, $P_k/Q_k = P/Q = s$.

1 Preliminaries

For a series in a single variable, a well-known theorem describes this result [9].

Theorem 1.1. *Let $\sum_{j=0}^{\infty} s_j X^{j+1} \in K[[X]]$ be a formal power series with coefficients in a field K of characteristic 0. Then there are 2 polynomials $P, Q \in K[X]$, such that*

$$deg(Q) < deg(P), \qquad \frac{Q}{P} = \sum_{j=0}^{\infty} \frac{s_j}{X^{j+1}}$$

if and only if there is an integer $p \in \boldsymbol{N}$ such that the ranks of the Hankel matrices at order k, $\forall k \geq p$, are p.
In that case, there exists P of degree p and Q of degree at most $p-1$. The minimal possible degree of P is p, and the pair (P, Q) is completely determined by these conditions of degree and on condition that P is monic. The polynomials P and Q are then prime.

The proof of this theorem is based on the resolution of a linear equations system obtained by identifying the coefficients of X^l.
Let us remark that the condition about the finiteness of the Hankel matrix rank of s expresses the recognizability of s, that is the rationality, for a single variable.

1.1 Commutative multivariate series

See [6].

Definition 1.2. K being a semiring, the set of the formal series $K[[X_i]]_{i\in I}$ built with the commutative variables $\{X_i\}_{i\in I}$ is made of infinite sums obtained from the commutative monoid $\{X_i\}^*_{i\in I}/C$, where $C = \{X_iX_j = X_jX_i \ \ \forall i \neq j\}_{i,j\in I}$.
The polynomials semiring is denoted by $K[X_i]_{i\in I}$.

Definition 1.3. Rational series in commutative variables

1. The set of the rational series $K[[X_i]]_{i\in I}$ is the smallest subring containing $K[X_i]_{i\in I}$ being rationally closed.
2. If K is a commutative field, a series $s \in K[[X_i]]_{i\in I}$ is rational if and only if s is the expansion at zero point of a rational fraction P/Q such that $P, Q \in K[X_i]_{i\in I}$ and $Q(0, \cdots, 0) \neq 0$.

Definition 1.4. Recognizable series in commutative variables

1. K being a commutative field, $s \in K[[X_i]]_{i\in I}$ is recognizable if and only if the K-module generated by the row-vectors or the column-vectors of its Hankel matrix, is free and finitely generated. The ranks of these 2 modules are equal to the rank of s.
   The Hankel matrix of $s \in K[[X_i]]_{i\in I}$ is defined as an infinite tabular, the rows and the columns of which are indexed by $\{X_i\}^*_{i\in I}/C$ and such that the element indexed by $(v, w) \in \{X_i\}^*_{i\in I}/C \times \{X_i\}^*_{i\in I}/C$ is the coefficient $\langle s|vw\rangle$.

2. K being a commutative field, $s \in K[[X_i]]_{i\in I}$ is recognizable if and only if s is the Taylor expansion at zero point of a rational fraction
   $$P/\prod_{j=1}^{k} Q_j$$
   where $P \in K[X_i]_{i\in I}$, and $Q_j \in K[X_j]$ such that $Q_j(0) \neq 0, \ \forall j,\ 1 \leq j \leq k$.

Unfortunately, For a series in several commutative variables the set of rational series strictly contains the set of recognizable series. For instance, the series $s \in \boldsymbol{R}[[X_1, X_2]]$
$$s = \sum_{n\geq 0} X_1^n X_2^n$$
is rational because it is the expansion of $\frac{1}{1-X_1X_2}$ but it is not recognizable because the rank of its Hankel matrix is infinite.

When we try to generalize the proof of theorem 1.1, for several commutative variables, the computations seem inextricable. An idea consists in using the computation in noncommutative variables.

1.2 Series in noncommutative variables

These definitions and notations are in [1, 13, 14, 15]. K is a semiring.

Definition 1.5. Formal power series in noncommutative variables

1. A non empty finite set X is an alphabet. An element of X is a letter.
   The free monoid X^* generated by the alphabet X is the set of the finite words $X_{i_1} \cdots X_{i_l}$ of some elements of X, including the empty word denoted by 1.
   The set X^* is a monoid for the concatenation product.

2. A formal power series s in noncommutative variables is a function:
   $$s : \ X^* \to K$$
   The coefficient $s(w)$ of w in s is denoted by $\langle s|w\rangle$.

3. The set of the formal power series s over X with coefficients in K is denoted by $K\langle\langle X\rangle\rangle$. A structure of semiring is defined on $K\langle\langle X\rangle\rangle$ by the sum and the Cauchy product. Two external operations (product acting on the left, on the right) from K to $K\langle\langle X\rangle\rangle$ are defined.
   The set of polynomials is denoted by $K\langle X\rangle$

1.2.1 Rational series in noncommutative variables

Definition 1.6. Rational formal power series in noncommutative variables

1. The rational operations in $K\langle\langle X\rangle\rangle$ are the sum, the product, the two external products on $K\langle\langle X\rangle\rangle$ and the star operation defined by:
   $T^* = \sum_{n\geq 0} T^n$ for T proper i.e. such that $\langle T|1\rangle = 0$

2. A subset of $K\langle\langle X\rangle\rangle$ is rationally closed if it is closed for the rational operations.
   The smallest subset containing a subset E of $K\langle\langle X\rangle\rangle$ which is rationally closed, is called the rational closure of E.

3. A series s is rational if s is element of the rational closure of $K\langle X\rangle$.

1.2.2 Recognizable series in noncommutative variables

We propose several equivalent definitions [1, 6, 7, 8, 12], K being a commutative field:

Definition 1.7. Recognizable formal power series in noncommutative variables

1. A series $s \in K\langle\langle X\rangle\rangle$ is recognizable if there exists an integer $N \geq 1$, and a monoid morphism
   $$\mu : X^* \to K^{N*N}$$
   and 2 matrices $\lambda \in K^{1*N}$ and $\gamma \in K^{N*1}$ such that
   $$\forall w \in X^*, \ \ \langle s|w\rangle = \lambda\mu(w)\gamma$$

2. A series $s \in K\langle\langle X\rangle\rangle$ is recognizable if there exists an integer N equal to the rank of its Hankel matrix $H(s) = (\langle s|w_1.w_2\rangle)_{w_1,w_2\in X^*}$.
   The first row of $H(s)$ indexed by the word 1 describes s.
   The other rows are the "remainders" of s by a word w. For instance, the row L_{X_1} represents the "right remainder" of s by X_1, denoted by $s \triangleright X_1$.

3. A series $s \in K\langle\langle X\rangle\rangle$ is recognizable if it is described by a finite weighted automaton obtained from its Hankel matrix "remainders".

1.2.3 Theorem of Schützenberger

For a series in several noncommutative variables, the theorem of Schützenberger proves the equivalence rationality-recognizability [15, 1].

Theorem 1.8. *A formal series is recognizable if and only if it is rational.*

1.2.4 Finite weighted automaton obtained from a rational series

This method is developed in [11]. It is based on the following theorem [8, 12]:

Theorem 1.9. *A formal series $s \in \boldsymbol{R}\langle\langle X\rangle\rangle$ is recognizable if and only if its rank N is finite. Then it is recognized by a $\boldsymbol{R}-$matricial automaton $M = (N, \gamma, \lambda, \mu)$. Two sets of words $\{g_i\}_{1\leq i\leq N}$ and $\{d_j\}_{1\leq j\leq N}$, the length of which is $< N$, can be determined such that the application χ from X^* to $\boldsymbol{R}^{N\times N}$ defined by*

$$(\chi(w))_{i,j} = \langle s|g_i.w.d_j\rangle$$

satisfies $\chi(w) = \chi(1)\mu(w)$ with $\chi(1)$ invertible.

1. The method consists in extracting, from its Hankel matrix $H(s)$ (the rank of which is N), a system B of N row-vectors $(L_{w_i})_{i\in I}$ (resp. N column-vectors $(C_{w_j})_{j\in J}$), indexed by some words of minimum length, such that their determinant is $\neq 0$ and such that every row (resp. every column) of $H(s)$ can be expressed as a linear combination of B.
   These relations allow us to define $\forall X_k \in X$ the matrices $\mu(X_k)$ describing the action of the letter X_k on the row-vector L_{w_i} (resp. the column-vector C_{w_j}).
   The first row (resp. the first column) of B defines λ.
   And γ is the initial vector ${}^t(1\ \ 0\cdots 0)$.
   So the series s can be written:

$$s = \sum_{w\in X^*} \langle s|w\rangle = \sum_{w\in X^*} \lambda\mu(w)\gamma$$

2. We define from B and matrices $\mu(X_i)$, γ and λ, a finite weighted (left or right) automaton $A = \{X, Q, I, A, \tau\}$ such that

   - X is the alphabet
   - The vertices set is $Q = \{L_{w_i}\}_{i\in I}$ representing $\{s \triangleright w_i\}_{i\in I}$(resp. $Q = \{C_{w_j}\}_{j\in J}$ representing $\{w_j \triangleleft s\}_{j\in J}$).
   - The first row (resp. the first column) I of B is the initial vertex.
   - Every transition between some vertices, belonging to τ is labeled by a letter $X_i \in X$ and valued by the coefficient appearing in the linear dependence relation.
   - A is the final states set. It is the set of the rows L_w (resp. the columns C_w) of B such that $\langle s|w\rangle \neq 0$.

1.2.5 Regular expression obtained from the finite *R*-weighted automaton

The method used is a generalization of the computing of the regular expression associated with a finite state automaton.

- We write the equations system satisfied by the automaton at every state $q_i \in Q$:

$$R_i = \sum_{j=1}^{card(Q)} (\sum_{k=1}^{n} \alpha_{i,j,k} X_k) R_j + \gamma_i, \quad \alpha_{i,j,k} \in \boldsymbol{R}$$

- We rewrite this system according to the following way:
  If $R_i = \Delta R_i + \Gamma$ satisfies the following hypothesis (H)

  Hypothesis (H) **:** Δ *and* Γ *are some regular expressions and* Δ *is proper*

  then $R_i = \Delta^* \Gamma$.
  By substituting this expression of R_i in every equation, we get a smaller equations system

$$R_j = \Lambda R_j + \Omega$$

  It is easy to prove that the equations system of the automaton satisfies the hypothesis (H) and that, after the suggested rewriting, these hypothesis are still satisfied.
  Then we obtain R_1 associated with the initial state, by making such substitutions. This is a presentation of the finite $\boldsymbol{R}$-weighted automaton as a regular expression.

And then, a rational series in noncommutative variables can be presented by a finite weighted automaton or by a regular expression [11].

Example 1.10. We take the alphabet $X = \{X_1, X_2\}$ and consider the series

$$s = \sum_{w \in X^*} (|w|_{X_1} - |w|_{X_2}) \ w$$

- Its Hankel matrix $H(s)$ is:

	1	X_1	X_2	X_1^2	X_1X_2	X_2X_1	$X_2^2 \cdots$
1	0	1	-1	2	0	0	$-2 \cdots$
X_1	1	2	0	3	1	1	$-1 \cdots$
X_2	-1	0	-2	1	-1	-1	$-3 \cdots$
X_1^2	2	3	1	4	2	2	$0 \cdots$
X_1X_2	0	1	-1	2	0	0	$-2 \cdots$
X_2X_1	0	1	-1	2	0	0	$-2 \cdots$
X_2^2	-2	-1	-3	0	-2	-2	$-4 \cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

- Its rank is 2. We choose the first two rows L_1, L_{X_1} associated with the first two columns C_1, C_{X_1} for generating $H(s)$.
  We have the following linear dependence relations:

$$\begin{array}{lcll} L_{X_2} & = & 2L_1 & -L_{X_1} \\ L_{X_1^2} & = & -L_1 & +2L_{X_1} \\ L_{X_1X_2} & = & L_1 & = L_{X_2X_1} \end{array}$$

  We extract $\mu(X_1)$, $\mu(X_2)$, λ et μ.

$$\mu(X_1) = \begin{pmatrix} 0 & -1 \\ 1 & 2 \end{pmatrix}$$

  and

$$\mu(X_2) = \begin{pmatrix} 2 & 1 \\ -1 & 0 \end{pmatrix}$$

  The initial vector is:

$$\gamma = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

  and the covector is:

$$\lambda = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

- A finite weighted automaton is represented by the graph given by the figure 1.

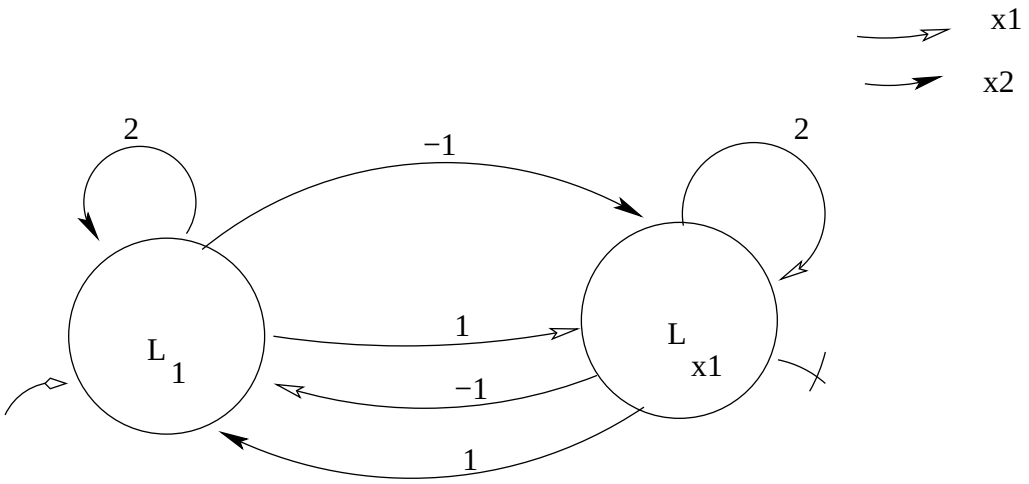


Figure 1: graph of the example 1.10

- We compute the regular expression associated with every state. Then we obtain:

$$s = (s \triangleright X_1)(X_1 - X_2) + s(2X_2)$$

$$(s \triangleright X_1) = s(X_2 - X_1) + (s \triangleright X_1)(2X_1) + 1$$

We extract from the second equation

$$s \triangleright X_1 = (s(X_2 - X_1) + 1)(2X_1)^*$$

We obtain by substituting in s:

$$s = (2X_1)^*(X_1 - X_2)[(X_2 - X_1)(2X_1)^*(X_1 - X_2) + 2X_2]^*$$

We write, by misusing the quotient:

$$s = \frac{1}{1 - 2X_1}(X_1 - X_2)\frac{1}{1 - (X_2 - X_1)\frac{1}{1-2X_1}(X_1 - X_2) - 2X_2}$$

This series is exchangeable (if we permute the order of the letters in the word, the coefficient is unchanged). Its commutative image (by setting $X_1X_2 = X_2X_1$) is:

$$s_{co} = \frac{X_1 - X_2}{(1 - (X_1 + X_2))^2}$$

Nevertheless this series is different from $s = \sum_{w \in X^*} (|w|_{X_1} - |w|_{X_2})\ w$
For instance, $\langle s_{co}|X_1^2X_2\rangle = 3$ because it is obtained by summing

$$\langle s|X_1^2X_2\rangle = 1,\ \ \langle s|X_1X_2X_1\rangle = 1,\ \ \langle s|X_2X_1^2\rangle = 1$$

2 Method for multivariate series $s \in \boldsymbol{R}[[X_i]]_{1 \leq i \leq n}$

We prove and use the following results.

2.1 Case of a recognizable series in commutative variables

Proposition 2.1. *If s is a recognizable series in commutative variables, the series $\underline{s_{nc}}$ in noncommutative variables obtained by assigning to 0 every coefficient except those of the increasing words $X_1^{i_1} \cdots X_n^{i_n}$ assigned to those of s, is recognizable.*

Proof. The proof consists in constructing from the Hankel matrix $H(s)$ the rank of which is finite, the Hankel matrix of $\underline{s_{nc}}$ and in proving that its rank is finite.
We use two steps

1. Let $\underbrace{s_{nc}}$ be the series in noncommutative variables obtained by assigning every word w such that $|w|_{X_1} = i$ and $|w|_{X_2} = j$ to $\langle s|X_1^iX_2^j\rangle$. This series is recognizable because its Hankel matrix is obtained by duplicating some rows and some columns from $H(s)$ and $rank(H(s)) = rank(H(\underbrace{s_{nc}}))$.

2. Let $\underbrace{A}$ be its finite weighted automaton containing n states, built with its Hankel matrix "remainders".

3. We construct from $\underbrace{A}$ a new automaton $\underline{A}$ associated with $\underline{s_{nc}}$ by suppressing every transition labeled by X_1 consecutive to some transitions labeled by X_2.
   The method is the following:

   - We make a copy of the part of $\underbrace{A}$ containing the transitions labeled by X_1 and the corresponding states. (If some transitions are both labeled by X_1 and X_2, we split them into two parts.) The states q_i are renamed q_{i1} in this copy.
   - We make a copy of the states which are an extremity of some transition labeled by X_2. These states q_j are renamed q_{j2} in this copy. We construct the transitions labeled by X_2 between q_{i1} and q_{j2}, labeled by X_2 between q_{j2} and q_{k2}, with the corresponding weigthing, if these transitions exist in $\underbrace{A}$ between the same states indexed with the first index.
   - q_{11} is the initial state of $\underline{A}$ if q_1 is the initial state of $\underbrace{A}$
     If q_t is a final state of $\underbrace{A}$ then if they exist, q_{t1} et q_{t2} are final states of $\underline{A}$.

   According to this construction, the automaton $\underline{A}$ contains at most $2n$ states.

And then the series $\underline{s_{nc}}$ is recognizable.

□

Corollary 2.2. *If s is recognizable, we can define a regular expression E describing $\underline{s_{nc}}$ and then*

$$s = commutative\_image(E) = P/Q$$

for some polynomials P, Q.

Example 2.3. Let us consider the recognizable series $s \in \boldsymbol{R}[[X_1, X_2]]$, satisfying

$$s = \sum_{i,j \geq 0} X_1^i X_2^j$$

According to Proposition 2.1, the rank of $\underline{s_{nc}}$ is finite.
We choose the basis made of the rows L_1, L_{X_2} associated with the columns C_1, C_{X_1} of its Hankel matrix. The rank is 2 ,

$$\mu(X_1) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

and

$$\mu(X_2) = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$$

The initial vector is:

$$\gamma = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

The covector is:

$$\lambda = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

A finite weighted automaton is represented by the graph given by the figure 2.

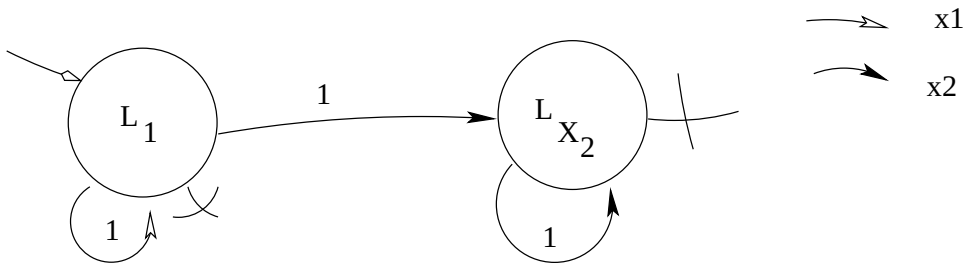


Figure 2: graph of the example 2.3

We obtain the regular expressions associated with the states L_1 and L_{X_2}:

$$\underline{s_{nc}} = 1 + X_1\underline{s_{nc}} + X_2(\underline{s_{nc}} \triangleright X_2)$$

$$\underline{s_{nc}} \triangleright X_2 = 1 + X_2(\underline{s_{nc}} \triangleright X_2)$$

We get

$$\underline{s_{nc}} \triangleright X_2 = (X_2)^*$$

And

$$\underline{s_{nc}} = (X_1)^*(1 + X_2X_2^*)$$

Then

$$s = \frac{1}{1 - X_1}\frac{1}{1 - X_2}$$

2.2 Case of a rational, non-recognizable series in commutative variables

Proposition 2.4. *If s is rational and non-recognizable in commutative variables, then s is commutative image of one (at least) series s_{nc} rational in noncommutative variables.*

Proof. Let $s \in \boldsymbol{R}[[X_i]]_{i\in I}$ be a rational series. According to the definition 1.3, there exists $P, Q \in \boldsymbol{R}[X_i]_{i\in I}$ such that $Q(0, \cdots, 0) \neq 0$ and

$$s = \frac{P(X_1, \cdots, X_n)}{Q(X_1, \cdots, X_n)}$$

By setting $Q(0, \cdots, 0) = q_0$, we obtain:

$$s = \frac{1}{q_0}\frac{P(X_1, \cdots, X_n)}{(1 - \frac{1}{q_0}R(X_1, \cdots, X_n))} = \frac{1}{q_0}\frac{P(X_1, \cdots, X_n)}{(1 - T(X_1, \cdots, X_n))}$$

with $R(X_1,\cdots,X_n), T(X_1,\cdots,X_n) \in \boldsymbol{R}[X_1,\cdots,X_n]$
The series

$$s_{nc} = \frac{1}{q_0} P_{nc}(X_1,\cdots,X_n)(T_{nc}(X_1,\cdots,X_n))^*$$

is just what we need, for some polynomials in noncommutative variables
$P_{nc}(X_1,\cdots,X_n), T_{nc}(X_1,\cdots,X_n) \in \boldsymbol{R}\langle X_1,\cdots,X_n\rangle$
such that
$P(X_1,\cdots,X_n), T(X_1,\cdots,X_n) \in \boldsymbol{R}[X_1,\cdots,X_n]$
are their respective commutative image.
The series s_{nc} is not generally single, P_{nc}, T_{nc} being generally multiple and the product of P_{nc} and T_{nc}^* in s_{nc} being possible at right like at left.

□

Corollary 2.5. *If s is rational and non-recognizable, we can progressively construct according to the length k of the words, the Hankel matrix $H(s_{nc_k})$ of a series s_{nc_k} of minimal rank, by maintaining the linear dependence relations existing in the already built part of the matrix (its rows and columns are ordered by increasing length of the words indexing them). Then*

$$\forall k, \;\; s_k = commutative\_image(s_{nc_k}) = P_k/Q_k$$

$\exists k_0 \in \boldsymbol{N}$ such that for $k \geq k_0$, there holds $s_k = P/Q = s$

Proof. For sake of simplicity, we present the proof in the case of 2 variables X_1, X_2. We order rows and columns of Hankel matrix by increasing length of the words indexing them and for a given length, the columns and the rows are ordered according to the lexicographic order of the words indexing them.
For a given order k, we build at first the part $H(s_{nc_k})_{\leq k}$ of the Hankel matrix $H(s_{nc_k})$ of s_{nc_k} related to the words of length $\leq k$. This matrix $\bar{H}(s_{nc_k})_{\leq k}$ will be then extended to the words of length $> k$ in an infinite matrix $H(s_{nc_k})$ of minimal rank.

1. Computing of $H(s_{nc_k})_{\leq k}$

   - Identification of coefficients of the words X_1^i, X_2^i:
     $\langle s_{nc_k}|X_j^i\rangle = \langle s|X_j^i\rangle \;\; \forall i \quad 0 \leq i \leq k, \quad j = 1,2$
   - Computing of all coefficients other than the previous ones

$$\sum_{|w|_{X_1}=p_1,\, |w|_{X_2}=p_2} \langle s_{nc_k}|w\rangle = \langle s|X_1^{p_1}X_2^{p_2}\rangle \qquad (E)$$

   The column C_1 is free. Two cases may occur:

– The column C_{X_1} is linearly dependent on C_1 in the already identified part. We extend it in order to preserve this linear dependence. By using (E), we obtain some coefficients such that $\langle s_{nc_k}|X_1X_2\rangle$ and we copy these new values in the columns C_{wX_1} and C_{wX_2}, for $|w| < k$. We consider now $C_{X_1X_2}, C_{X_2^2}$, the columns C_{wX_1} being necessarily dependent on the previous ones.
– The column C_{X_1} is free in the already identified part. We consider C_{X_2}. If it is dependent on the previous columns C_1, C_{X_1} in the already identified part, we extend it in order to preserve this linear dependence and we copy the new values in the columns C_1, C_{wX_2} and, by using (E), in C_{wX_1}.
If C_{X_2} is free, we consider the next column $\cdots$

So we obtain $H(s_{nc_k})_{\leq k}$, without getting necessarily every identified coefficient. Generally, it remains some indeterminates by extending linearly the columns.

2. Extension of $H(s_{nc_k})_{\leq k}$ to $H(s_{nc_k})$ (see [10])
We extend $H(s_{nc_k})_{\leq k}$ to the words of length $> k$ by maintaining the linear relations existing in the already built part. So we obtain $H(s_{nc_k})$, Hankel matrix of s_{nc_k}, noncommutative rational series of minimal rank such that its commutative image s_k satisfies:

$$order(s - s_k) > k$$

□

Example 2.6. Let $s \in \boldsymbol{R}[[X_1, X_2]]$ be the series:

$$s = \sum_{i,j \in \boldsymbol{N}} X_1^i X_2^j \Big(\sum_{k=0}^{inf(i,j)} \begin{pmatrix} |i-j| + 2k \\ k \end{pmatrix} \Big)$$

This series is rational and we will obtain $rank(s_{nc}) = 3$.
For $k = 2$, the Hankel matrix $H(s_{nc_2})_{\leq 2}$ get by the previous algorithm is

	1	X_1	X_2	X_1^2	X_1X_2	X_2X_1	X_2^2
1	1	1	1	1	$2*$	$1*$	1
X_1	1	1	$2*$				
X_2	1	$1*$	1				
X_1^2	1						
X_1X_2	$2*$						
X_2X_1	$1*$						
X_2^2	1						

where the values marked by a star are computed according to the algorithm. The rank is 2, a basis being $\{C_1, C_{X_2}\}$.
Then we can provide $s_{nc_2} = (X_1 + 2X_2X_2^*X_1)^*(1 + X_2X_2^*)$ and $s_2 = \frac{1}{1-(X_1+X_2+X_1X_2)}$.

For $k \geq 4$, the rank of s_{nc_k} is 3.
For instance, for $k = 4$, the Hankel matrix $H(s_{nc_4})_{\leq 4}$ get by the previous algorithm is

	1	X_1	X_2	X_1^2	X_1X_2	X_2X_1	X_2^2	X_1^3	$\cdots$	$X_1^2X_2$	$X_1X_2^2$	$X_2X_1X_2$	X_2^3	$\cdots$
1	1	1	1	1	2	1	1	1	$\cdots$	1	2	1	1	$\cdots$
X_1	1	1	2	1	1	1	2	1	$\cdots$	1	1	3	2	$\cdots$
X_2	1	1	1	1	1	1	1	1	$\cdots$	1	1	1	1	$\cdots$
X_1^2	1	1	1	1	1	1	1							$\cdots$
X_1X_2	2	2	2	2	3	2	2							$\cdots$
X_2X_1	1	1	1	1	1	1	1							$\cdots$
X_2^2	1	1	1	1	1	1	1							$\cdots$
X_1^3	1	1	1											$\cdots$
$\cdots$														$\cdots$
$X_1^2X_2$	1	1	1											$\cdots$
$X_1X_2^2$	2	2	2											$\cdots$
$X_2X_1X_2$	1	1	1											$\cdots$
X_2^3	1	1	1											$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

We choose the basis $\{C_1, C_{X_2}, C_{X_1X_2}\}$ and we obtain:

$$s_{nc_4} = s_{nc} = [X_1 + X_2(X_2 + 2X_1X_2)^*X_1(X_1 - X_2)]^*[1 + X_2(X_2 + 2X_1X_2)^*(1 + 2X_1)]$$

Its commutative image is:

$$s_k = s = \frac{1}{(1 - X_1X_2)(1 - (X_1 + X_2))}$$

Particular cases

1. The noncommutative series s_{nc} is obtained by evenly distributing the coefficient of $X_1^iX_2^j$ in s (by symmetry) over every words having this same commutative image $X_1^iX_2^j$.

   Example 2.7.

   $$s = \sum_{n \geq 0} (X_1 + X_2)^n$$

   Then $s_{nc} = \sum_{w \in \{X_1, X_2\}^*} w$ and

   $$s = \frac{1}{1 - (X_1 + X_2)}$$

2. The noncommutative series s_{nc} is obtained by assigning to 0 every coefficient of w in s_{nc} except $\forall \ \ i, j$ the coefficient of a single word $w_{i,j}$ containing i occurences of X_1 and j occurences of X_2 assigned to the coefficient of $X_1^iX_2^j$ in s.

   Example 2.8.

   $$s = \sum_{n \geq 0} X_1^nX_2^n$$

   Then $s_{nc} = (X_1X_2)^*$ and

   $$s = \frac{1}{1 - X_1X_2}$$

2.3 Case of an ordinary multivariate series

Corollary 2.9. *For every multivariate series $s \in \boldsymbol{R}[[X_i]]_{1\leq i\leq n}$, we can construct a family $(s_k)_{k\in\boldsymbol{N}}$ of rational series such that the difference $s - s_k$ is at least of order k.*

3 Conclusion

For a given multivariate series $s \in \boldsymbol{R}[[X_i]]_{1\leq i\leq n}$, we propose a new algorithm providing a family of rational approximants $(P_k/Q_k)_{k\in\boldsymbol{N}}$, r_k being the minimal rank of the noncommutative intermediate series s_{nc_k}.
The disadvantages of this method are the following: From order k to order $k+1$, the computing is not generally acquired. There is not generally unicity of s_{nc_k}. Moreover, the complexity of this algorithm is acceptable for $n = 2$ variables, the number of words of length $\leq k$ built on an alphabet of n letters being $O(n^k)$.
The advantages of this method are: The computations are elementary. The rational approximants provided s_{nc_k} are of minimal rank r_k. Moreover, when this rank is small, the method is all the more efficient because the basis vectors are indexed by short words.
And when s is rational, the family $(P_k/Q_k)_{k\in\boldsymbol{N}}$ is stationary and there is an order k_0 such that $\forall k \geq k_0$, $P_k/Q_k = P/Q = s$.
This algorithm of rational approximation is suitable for implementation in a symbolic environment.

References

[1] Berstel J., Reutenauer C., *Rational series and their languages*, Springer-Verlag, 1988.

[2] Cuyt A., *A comparison of some multivariate Padé approximants*, SIAM J. Math. Anal.,vol. 14, pp. 195-202, 1983.

[3] Cuyt A., *How well can the concept of Padé approximant be generalized to the multivariate case*, J. Comput. Appl. Math., vol. 105, pp. 25-50, 1999.

[4] Drosde M., Gastin P., *The Kleene-Schützenberger theorem for formal power series in partially commuting variables*, Information and Computation, vol.153, pp. 47-80, 1999.

[5] Fitzpatrick P., Flynn J., *A Gröbner basis technique for Padé approximation*, J. Symbolic Comput., vol 13, pp. 133-138, 1992.

[6] Fliess M., *Sur certaines familles de séries formelles*, Thèse d'état, Université de Paris 7, 1972.

[7] Fliess M., *Matrices de Hankel*, J. Maths. Pur. Appl., vol.53, pp. 197-222, 1974.

[8] Fliess M., *Un outil algébrique : les séries formelles non commutatives*, in "Mathematical System Theory" (G. Marchesini and S.K. Mitter Eds.), Lecture Notes Econom. Math. Syst., Springer Verlag, vol. 131, pp. 122-148, 1976.

[9] Gantmacher F.R., *Théorie des matrices*, tome 2, Dunod, 1966.

[10] Hespel C., Jacob G., *Approximation of nonlinear dynamic systems by rational series*, Theoret. Comput. Sciences, vol. 79, pp. 151-162, 1991.

[11] Hespel C., *Une étude des séries formelles non commutatives pour l'Approximation et l'Identification des systèmes dynamiques*, Thèse d'état, Université de Lille 1, 1998.

[12] Jacob G., *Réalisation des systèmes réguliers (ou bilinéaires) et séries génératrices non commutatives*, Séminaire d'Aussois, RCP567, Outils et modèles mathématiques pour l'Automatique, l'Analyse des Systèmes, et le traitement du Signal (CNRS Landau), 1980.

[13] Reutenauer C., *Séries formelles et algèbres syntactiques*, J. Algebra, vol. 66, pp. 448-483, 1980.

[14] Salomaa A., Soittola M., *Automata Theoretic aspects of Formal Power Series*, Springer, 1978.

[15] Schützenberger M.P., *On the definition of a family of automata*, Inform. and Control, vol. 4, pp. 245-270, 1961.

Christiane Hespel
IRISA-INSA de Rennes
35043 Rennes cedex, France
hespel@irisa.fr

Cyrille Martig
IRISA-INSA de Rennes-Ecoles de Coëtquidan
35043 Rennes cedex, France
cyrille_martig@yahoo.fr

Self-similar trajectories in multi-input systems

Roland Hildebrand

Abstract

Self-similar trajectories play an important role in deterministic feedback control systems that possess a symmetry group of Fuller type. We consider self-similar trajectories in multi-input systems and the linear part of the associated Poincaré maps in orbit space with respect to the symmetry group. We show that Fuller groups contract the symplectic structure of the system's phase space and derive some properties of the spectrum and the eigenvectors of the Poincaré map.

Introduction

We consider self-similar trajectories in deterministic feedback control systems, which give rise to Hamiltonian dynamics via Pontryagins maximum principle. Self-similar trajectories were discovered by Fuller in 1960 [1]. Fuller considered the optimal control system

$$\dot{x} = y, \quad \dot{y} = u \in [-1, 1], \quad x(0) = x_0, \quad y(0) = y_0, \quad J(u(\cdot)) = \frac{1}{2}\int_0^\infty x^2(t)\,dt \to \inf. \quad (1)$$

Here u is a bounded scalar control; x, y parametrize the state space $\mathbb{R}^2$; and the cost functional J is to be minimized over all measurable control functions $u(t)$. Fuller has found that outside the origin the optimal feedback control is bang-bang. The origin itself is a singular stationary trajectory of second order. The ratio of successive intervals of time between successive control switchings turned out to be constant. Therefore Fuller named the corresponding trajectories *constant-ratio trajectories*. They are the prototype of self-similar trajectories. In system (1) the origin is an accumulation point of control switchings. This phenomenon is called *chattering* and is closely related to self-similar trajectories.

Fuller discovered [2] that system (1) possesses a one-parametric symmetry group $\mathcal{G}_F$. Namely, if $(x(t), y(t), u(t))$ is an optimal trajectory of the system, then $(\lambda^2 x(\frac{t}{\lambda}), \lambda y(\frac{t}{\lambda}), u(\frac{t}{\lambda}))$ is also an optimal trajectory for any positive number λ. We say that a trajectory is *self-similar* if it is invariant with respect to a nontrivial subgroup of $\mathcal{G}_F$. Consider an orbit of $\mathcal{G}_F$ in state space. The phase velocity of the optimal synthesis has the same direction at any point of this orbit, and determines a direction field on orbit space. Without the origin $(0, 0)$, the orbit space will be homeomorphic to S^1. It consists of a single periodic orbit of the direction field and corresponds to a one-parametric family of self-similar trajectories.

This example demonstrates that self-similar trajectories are important elements in the structure of the optimal synthesis of systems with a symmetry of Fuller type. Namely, they

correspond to periodic orbits and stationary points of the direction field that is defined on orbit space with respect to the symmetry group. Fuller [3] developped a detailed algorithm for calculating self-similar trajectories in the case of a symmetrically bounded scalar input. He considered the case where the control jumps between the two extremal values and the intervals between successive control switchings form a decreasing geometric progression.

The techniques presented in this paper allow to find self-similar trajectories in the general case and to reveal the behaviour of the system in the neighbourhood of these trajectories.

Applications of self-similar trajectories are not restricted to systems with a Fuller group. Zelikin and Borisov developped a theory of chattering [7]. In particular, they showed that, provided a certain approximate Fuller symmetry holds, singular manifolds of codimension 2 and order 2 serve as base of a fibration in state space with fibres orbitally equivalent to the optimal synthesis of system (1). The global structure of the synthesis is retained in the fibre, and the self-similar trajectories of the non-perturbed system possess equivalents in the perturbed system, although the latter are no more self-similar.

The paper is organized as follows. In section 1 we define self-similar trajectories and Fuller groups and describe the treated class of optimal control systems. In sections 2 and 3 we develop algorithms for finding self-similar trajectories and calculating the linear part of the Poincaré map associated with the corresponding periodic orbits, or alternatively, the linearization in the neighbourhood of corresponding stationary points (Algorithms 1 and 2, Theorem 2.1). We provide a criterion of optimality of a self-similar trajectory (Theorem 2.4) and investigate integral manifolds consisting of optimal trajectories in a neighbourhood of a hyperbolic periodic orbit (Theorem 3.3). In the next section we show that Fuller groups contract the sympletic structure of phase space and derive some properties of the spectrum and the eigenvalues of the Poincaré map (Theorems 4.10 and 4.11). Finally we draw some conclusions.

1 Definitions and preliminaries

We look for self-similar trajectories in systems given by the following generalization of (1).

$$x = (x_1, \dots, x_l) \in \mathbb{R}^l,\ \sum_{i=1}^{l} m_i x_i^{(j)} = 0\ \forall\ j = 1, \dots, n-1;\ J(u(\cdot)) = \frac{1}{2}\int_0^\infty \sum_{i=1}^{l} m_i x_i^2\, dt \to \inf;$$

$$x^{(n)} = u, \quad u \in U = \left\{ (u_1, \cdots, u_l) \in \mathbb{R}^l \,|\, u_i \le 1\ \forall\ i = 1, \dots, l;\ \sum_{i=1}^{l} m_i u_i = 0 \right\}. \qquad (2)$$

Here $m_i > 0$; the state space M is parametrized by the l components of the vector x and its time derivatives $x_i^{(j)}$ up to order $n-1$. Since there are restrictions $\sum_{i=1}^{l} x_i^{(j)} = 0$, the state space has $n \times (l-1)$ dimensions. The cost functional J is to be minimized over all measurable inputs $u(t)$. The set U of admissible control values is an $(l-1)$-dimensional simplex embedded in the space $\mathbb{R}^l$ parametrized by the coordinates $u_1, \dots, u_l$. Suppose the initial point $(x(0), \dot{x}(0), \dots, x^{(n-1)}(0))$ in state space is fixed; then the functional J is

strictly convex on the convex set of all admissible functions $u(\cdot)$. Hence by Kuhn-Tucker's theorem, the optimal solution exists and is unique for any fixed initial point in state space.

Let us apply Pontryagin's maximum principle [6] to system (2). We obtain the Pontryagin function $H = -\frac{1}{2}\sum_{i=1}^{l} m_i x_i^2 + \sum_{j=1}^{n-1}\sum_{i=1}^{l} \psi_i^j x_i^{(j)} + \sum_{i=1}^{l} u_i \psi_i^n$. Here the vector $\psi^j = (\psi_1^j, \dots, \psi_l^j)$ is conjugated to the vector of $(j-1)$-th order derivatives $x^{(j-1)}$. The conjugated variables satisfy the equations $\dot\psi_i^1 = m_i x_i \ \forall\, i = 1, \dots, l$; $\dot\psi^j = -\psi^{j-1} \ \forall\, j = 2, \dots, n$. The control u is determined by ψ^n according to the maximum principle. Suppose there exists a $i \in \{1, \dots, l\}$ such that $\frac{\psi_i^n}{m_i} < \frac{\psi_{i'}^n}{m_{i'}}$ for any $i' \neq i$, then the control is given by

$$u_{i'} = 1 \ \forall\ i' \neq i, \qquad u_i = -\frac{\sum_{i' \neq i} m_{i'}}{m_i}. \tag{3}$$

Without loss of generality we can put $\sum_{i=1}^{l} \psi_i^j = 0$ for all $j = 1, \dots, n$ [4]. We consider only regular trajectories in system (2). By [4, Remark on p.3] we then cover also the case of singular trajectories, anyway.

Let us introduce coordinates y_i^j in phase space; here $i \in \{1, \dots, l\}, j \in \{1, \dots, 2n\}$:

$$y_i^j = (-1)^{j-1}\frac{\psi_i^{n-j+1}}{m_i}, \quad y_i^{n+j} = (-1)^{n-1} x_i^{(j-1)}, \quad \forall\ i = 1, \dots, l; j = 1, \dots, n.$$

Let us join coordinates with equal upper indices to vectors $y^j \in \mathbb{R}^l$. Then the restrictions and Hamiltonian dynamics are given by

$$\dot y^j = y^{j+1} \ \forall\ j = 1, \dots, 2n-1; \quad \dot y^{2n} = (-1)^{n-1} u; \qquad \sum\nolimits_{i=1}^{l} m_i y_i^j = 0 \ \forall\ j = 1, \dots, 2n. \tag{4}$$

The control u is given by (3); here i is the index for which y_i^1 is smallest. The phase space of the system is parametrized by the $2nl$ components of the vectors $y^1, \dots, y^{2n}$ with $2n$ restrictions. Hence it has $2n(l-1)$ dimensions. The phase space can be considered as the cotangent fibration T^*M over the state space M. The latter is parametrized by the components of the vectors $y^{n+1}, \dots, y^{2n}$ with n restrictions. It has $n(l-1)$ dimensions.

Proposition 1.1. *[4, Proposition 1] Suppose ρ is a trajectory of system (4). Then ρ is a lifting to T^*M of an optimal trajectory if and only if it tends to the origin of T^*M.*

In the sequel, we shall call any trajectory in phase space T^*M optimal, if it is a lift of an optimal trajectory in state space M.

On phase space there acts a one-parametric group $\mathcal{G}$ of linear transformations. It is parametrized by a multiplicative parameter $\lambda \in \mathbb{R}_+$. The element $\mathcal{G}_\lambda$ corresponding to the number λ multiplies the coordinate y_i^j by λ^{2n-j+1}, $\mathcal{G}_\lambda : (y^1, \dots, y^{2n}) \mapsto (\lambda^{2n} y^1, \dots, \lambda y^{2n})$.

Definition 1.2. We call the group $\mathcal{G}$ *Fuller group.*

The action of $\mathcal{G}$ on T^*M induces an action on state space M, because the action on the variables $y^{n+1}, \dots, y^{2n}$ does not depend on the values of the other variables. It is readily seen that $\mathcal{G}_\lambda$ takes any trajectory of system (4) to another trajectory with a change of time scale by the factor λ. Optimal trajectories are taken to optimal trajectories.

Corollary 1.3. *Suppose $y(t)$ is a trajectory of system (4); then for any $\lambda > 0$ the trajectory $y'(t) = \mathcal{G}_\lambda(y(\frac{t}{\lambda}))$ is also a solution of system (4). If $y(t)$ is optimal, then so is $y'(t)$.* □

Thus system (4) induces a direction field on orbit space $(T^*M)/\mathcal{G}$. If we remove the origin, which is a separate orbit, orbit space will be homeomorphic to the sphere $S^{2n(l-1)-1}$. Denote the space $((T^*M)/\mathcal{G}) \setminus \{0\} \cong S^{2n(l-1)-1}$ by Σ^*. Since at any point of an orbit the same control is applied, the control function u given by (3) is defined also on Σ^*. We have the same situation in state space. The velocity field defined on M by the optimal synthesis points in the same direction at any point of a given orbit with respect to $\mathcal{G}$. Hence the optimal synthesis induces a direction field on orbit space $M/\mathcal{G}$. If we remove the origin, then this orbit space will be homeomorphic to the sphere $S^{n(l-1)-1}$. Denote the space $(M/\mathcal{G}) \setminus \{0\} \cong S^{n(l-1)-1}$ by Σ. The optimal synthesis defines a section $M \to T^*M$ of the cotangent fibration. This section induces an embedding $\Sigma \to \Sigma^*$. In the sequel, we will identify Σ with its image in Σ^*. In view of the above, we call a trajectory in Σ^* an optimal trajectory if it belongs to Σ.

Definition 1.4. We call any trajectory of system (4) *self-similar*, if it is invariant with respect to a non-trivial subgroup of $\mathcal{G}$.

Corollary 1.5. *The preimage of a self-similar trajectory in Σ^* consists of stationary points and/or periodic orbits.* □

Definition 1.6. We call a periodic orbit ζ in Σ^* an *s-chain*, if the restriction of the control function to ζ is piecewise constant and has s points of discontinuity.

It can be shown that any periodic orbit in Σ is an s-chain for some $s > 1$. Hence we can restrict our study to stationary points and s-chains consisting of regular arcs.

2 An algorithm for finding self-similar trajectories

Beside the group $\mathcal{G}$, there exists a subgroup of the permutation group S_l that acts on state space of system (4). This group permutes components y_i^j of the vectors y^j that correspond to equal weights m_i. Let us denote this subgroup by $\mathcal{S}$. Clearly it takes optimal trajectories to optimal trajectories. Moreover, the action of $\mathcal{S}$ commutes with the action of $\mathcal{G}$. Hence, $\mathcal{S}$ acts also on the spaces Σ and Σ^*.

There exists an involution $\mathcal{T}$ of system (4), which multiplies the coordinate y_i^j by $(-1)^{j+1}$. Suppose $y(t)$ is a trajectory of system (4); then $y'(t) = \mathcal{T}(y(-t))$ is also a trajectory. However, $\mathcal{T}$ does not take optimal trajectories to optimal ones.

We look for periodic orbits in Σ^* that have a finite number of control switchings and consist of regular arcs. Let us index the l vertices of the simplex U of admissible controls. We denote control (3) by *control i*, $i = 1, \ldots, l$.

Theorem 2.1. *Self-similar trajectories with constant control exist only for odd n. In this case for any $i \in \{1, \ldots, l\}$ there is a unique self-similar trajectory with constant control i. This trajectory passes through the origin and consists of three orbits of the group $\mathcal{G}$. One of them corresponds to a stationary point on Σ.*

The proof is by integrating the system dynamics and applying Proposition 1.1.

Proposition 2.2. *If n is even, then there are no stationary points on Σ. If n is odd, then there exist exactly 2^l-2 stationary points on Σ. l of them correspond to regular trajectories.*

Proof. There is a bijection between stationary points on Σ and optimal self-similar trajectories with constant control. By Theorem 2.1, there are no such trajectories for even n. Suppose n is odd. It is not hard to deduce from the maximum principle that any proper face of the simplex U contains a unique control that is realized on an optimal trajectory with constant control. But there are $2^l - 2$ such faces. The regular trajectories correspond to the l vertices of the simplex. □

Consider an s'-chain ζ ($s' > 1$), where the controls $i_1, \dots, i_{s'}$ are used successively, thereafter the chain closes. Denote the initial switching point of ζ by $\tilde{q}_0$. Denote the next switching point, which is attained after using control i_1, by $\tilde{q}_1$, the next by $\tilde{q}_2$ and so on, up to the terminal point $\tilde{q}_{s'}$, which coincides with $\tilde{q}_0$. Consider a point $q_0 \in T^*M$ on the orbit $\tilde{q}_0$. Denote the trajectory of system (4) that goes through q_0 by ρ. Denote the next switching point on ρ by q_1, the next by q_2 and so on, up to $q_{s'}$. The point $q_{s'}$ lies again on the orbit $\tilde{q}_{s'} = \tilde{q}_0$, but in general it does not coincide with q_0. There exists exactly one positive number λ' such that the element $\mathcal{G}_{\lambda'} \in \mathcal{G}$ takes q_0 to $q_{s'}$.

Definition 2.3. We call the number λ' the *contraction coefficient* of the s'-chain ζ.

The definition is correct, because λ' does not depend on the initial point $q_0 \in \tilde{q}_0$. By Proposition 1.1, the following assertions hold.

Theorem 2.4. *An s-chain with contraction coefficient λ is optimal if and only if $\lambda < 1$.* □

Corollary 2.5. *Trajectories in Σ^* that converge to on optimal s'-chain ζ are optimal and lie in the space Σ.* □

The complexity of the equations determining $\tilde{q}_0$ and λ' quickly increases with s'. But if we restrict our investigations to chains that are invariant with respect to an element of the discrete symmetry group $\mathcal{S}$, then the number of relevant switchings can be reduced.

Suppose there exists a divisor s of s' and a permutation $\sigma \in \mathcal{S}$ such that σ takes $\tilde{q}_0$ to $\tilde{q}_s$. There exists a number λ such that $\mathcal{G}_\lambda(\sigma(q_0)) = q_s$. Clearly $\lambda' = \lambda^{s'/s}$, and the condition $\lambda' < 1$ is equivalent to $\lambda < 1$. Instead of the equation $\mathcal{G}_{\lambda'}(q_0) = q_{s'}$ we can now consider the simpler equation $\mathcal{G}_\lambda(\sigma(q_0)) = q_s$, for fixed σ and $i_1, \dots, i_s$.

By definition, put $i_0 = \sigma^{-1}(i_s)$ and $i_{s+1} = \sigma(i_1)$. For any $k \in \{1, \dots, s\}$ the point q_k lies on the orbit $\tilde{q}_k$. On the arc that connects q_{k-1} and q_k the control i_k is applied. On the interior of this arc the condition of optimality of control i_k holds:

$$y^1_{i_k} < y^1_{i_{k'}} \ \forall\ i_{k'} \neq i_k. \tag{5}$$

Denote the time that system (4) needs to pass from q_{k-1} to q_k by t_k. Let us compute q_k as a function of q_{k-1} by integrating (4) with control i_k. Assemble the coordinates of phase space in a vector $y = (y_1^{2n}, \dots, y_l^{2n}, y_1^{2n-1}, \dots, y_l^1)^T \in \mathbb{R}^{2nl}$. Then we can rewrite system (4) as $\dot{y} = Ay + b_{i_k}$. Here A is a constant matrix. The diagonal that is obtained by shifting the main diagonal by l positions downwards is filled with 1's, the rest with zeros. The constant vector b_i depends only on the control i. Its first l elements form the control i multiplied by

$(-1)^{n-1}$, the other elements are zero. The matrix exponent $F(t) = e^{tA}$ is

$$F(t) = \begin{pmatrix} D_0 & 0 & \dots & 0 \\ D_1 & D_0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ D_{2n-1} & D_{2n-2} & \dots & D_0 \end{pmatrix}.$$

Here $D_i = \frac{t^i}{i!} I_l$ and I_l is the $l \times l$ identity matrix. By integrating the system with initial value q_{k-1} we obtain $y(q_k) = F(t_k)y(q_{k-1}) + \bar{B}_k$. Here $\bar{B}_k$ depends only on t_k and b_{i_k},

$$\bar{B}_k = \int_0^{t_k} F(t_k - \tau) b_{i_k} \, d\tau = C(t_k) \int_0^1 F(1-\tau) b_{i_k} \, d\tau = C(t_k) B_{i_k}.$$

Here $C(t) = \mathrm{diag}(tI_l, t^2 I_l, \dots, t^{2n} I_l)$ is a diagonal $2nl \times 2nl$-matrix, and $B_i = \int_0^1 F(1-\tau) b_i \, d\tau$ is a constant vector, which depends only on control i.

By iterating the equation $y(q_k) = F(t_k)y(q_{k-1}) + \bar{B}_k$ we finally obtain

$$y(q_k) = F(\textstyle\sum_{i=1}^{k} t_i) y(q_0) + \sum_{j=1}^{k} F(\sum_{i=j+1}^{k} t_i) \bar{B}_j. \tag{6}$$

By definition empty sums are zero. Denote the linear operator corresponding to the permutation σ by H_σ. The action of the element $\mathcal{G}_\lambda$ can be described as multiplication by the matrix $C(\lambda)$. Hence we get the polynomial system of equations

$$y(q_s) = y(\mathcal{G}_\lambda(\sigma(q_0))) = C(\lambda) H_\sigma y(q_0) = F(\textstyle\sum_{i=1}^{s} t_i) y(q_0) + \sum_{j=1}^{s} F(\sum_{i=j+1}^{s} t_i) \bar{B}_j. \tag{7}$$

Further, $F(t)$ is block-triangular with identity matrices on the diagonal. Since eigenvalues of permutation matrices have absolute value 1, we get

Proposition 2.6. *Suppose* $|\lambda| \neq 1$, *then* $\det(C(\lambda)H_\sigma - F(\sum_{i=1}^{s} t_i)) \neq 0$. □

For optimal chains we have $|\lambda| \neq 1$, and equations (7) can be resolved with respect to $y(q_0)$:

$$y(q_0) = \left(C(\lambda) H_\sigma - F(\textstyle\sum_{i=1}^{s} t_i) \right)^{-1} \left(\sum_{j=1}^{s} F(\sum_{i=j+1}^{s} t_i) \bar{B}_j \right). \tag{8}$$

The point q_k lies on the switching hypersurface from control i_k to control i_{k+1}. This implies $y^1_{i_k}(q_k) = y^1_{i_{k+1}}(q_k)$. By v_k denote the $2nl$-dimensional row vector that has a 1 on position $((2n-1)l + i_k)$ and a -1 on position $((2n-1)l + i_{k+1})$, all other elements being zero. v_k depends only on the controls i_k and i_{k+1}. Then $y^1_{i_k}(q_k) = y^1_{i_{k+1}}(q_k)$ transforms into

$$v_k \left[F(\textstyle\sum_{i=1}^{k} t_i) y(q_0) + \sum_{j=1}^{k} F(\sum_{i=j+1}^{k} t_i) \bar{B}_j \right] = 0. \tag{9}$$

For $k = 0, \dots, s-1$ we obtain a system of s polynomial equations. It is easily checked that equation (9) for $k = s$ follows from equation (9) for $k = 0$ and equation (7).

Equations (7) and (9) form an overdetermined linear system of $2nl + s$ equations with respect to the $2nl$ unknown components of the vector $y(q_0)$. Let us join the coefficient matrix of this system and its right-hand side to the $(2nl + s) \times (2nl + 1)$-matrix

$$L = \begin{pmatrix} C(\lambda)H_\sigma - F(\sum_{i=1}^{s} t_i) & \sum_{j=1}^{s} F(\sum_{i=j+1}^{s} t_i)\bar{B}_j \\ v_0 & 0 \\ v_1 F(t_1) & -v_1 \bar{B}_1 \\ \vdots & \vdots \\ v_{s-1} F(\sum_{i=1}^{s-1} t_i) & -v_{s-1}\left(\sum_{j=1}^{s-1} F(\sum_{i=j+1}^{s-1} t_i)\bar{B}_j\right) \end{pmatrix}. \tag{10}$$

System (7), (9) has a solution if and only if all minors of dimension $2nl+1$ of L vanish. By Proposition 2.6, this is equivalent to vanishing of all s minors that contain the first $2nl$ rows. Thus we obtain a system of s polynomial equations on $s + 1$ unknown quantities $t_1, \dots, t_s; \lambda$. Since the point q_0 on the orbit $\tilde{q}_0$ can be chosen arbitrarily, we can introduce an additional condition, e.g. $\sum_{i=1}^{s} t_i = 1$. Then the number of unknowns will be equal to the number of equations. Any solution that satisfies conditions (5) and the inequalities $t_i > 0 \ \forall \ i = 1, \dots, s; \ \lambda > 0, \lambda \neq 1$ determines an s'-chain on Σ^*. The coordinates of the point q_0 are to be found from (8). It can easily be seen that the restrictions on y_i^j are fulfilled. On ζ the a priori chosen sequence of controls $i_1, \dots, i_s$ is realized and ζ is invariant with respect to the symmetry σ. Theorem 2.4 yields that ζ is optimal if $\lambda < 1$.

Any minor of L that contains the first $2nl$ rows is a homogeneous polynomial of degree $2n$ with respect to $t_1, \dots, t_s$ and a polynomial of degree $nl(2n+1)$ with respect to λ. Hence the degree of the polynomials does not depend on the parameters s, σ, $i_1, \dots, i_s$.

Thus we obtain the following algorithm for detection of chains with a given sequence of controls and invariance with respect to a given symmetry $\sigma \in \mathcal{S}$. Any solution corresponds to a one-parametric family of self-similar trajectories related by the action of $\mathcal{G}$.

Algorithm 1: *(compare also [4, Section 3]) Calculation of optimal self-similar trajectories.*

1. Calculate the matrix L according to (10).

2. Compose a system of polynomial equations by putting all minors of dimension $2nl+1$ of L to zero that contain the first $2nl$ rows.

3. Solve the system with an additional constraint eliminating homogeneity in the t_i.

4. Calculate the coordinates of the initial point q_0 of the self-similar trajectory according to (8), and of the other switching points according to (6).

5. Check the conditions $\lambda \in (0, 1), t_i > 0$ and (5).

A similar algorithm was given in [3] for the case $l = 2$, $m_1 = m_2 = 1$, $s = 1$, $\sigma = (12)$, n arbitrary, and in [5] for the case $l = 2$, $n = 2$, $s = s' = 2$, m_i arbitrary.

3 The linear part of the Poincaré map

Let ζ be an s'-chain, invariant with respect to the permutation $\sigma \in \mathcal{S}$, with parameters $t_1, \dots, t_s; \lambda$ and switching points $\tilde{q}_0, \dots, \tilde{q}_s$. Let us investigate the behaviour of the dynam-

ical system on Σ^* in a neighbourhood of ζ. We consider only the generic case, when ζ transversally intersects the switching hypersurfaces at the switching points and the switching points do not lie on the intersection of more than one switching hypersurface. Then

$$y^1_{i_k}(q_k) = y^1_{i_{k+1}}(q_k) < y^1_{i_{k'}}(q_k) \;\forall\; i_{k'} \notin \{i_k, i_{k+1}\};\; \dot{y}^1_{i_k}(q_k) - \dot{y}^1_{i_{k+1}}(q_k) = y^2_{i_k}(q_k) - y^2_{i_{k+1}}(q_k) > 0. \tag{11}$$

Let us consider a neighbourhood $\tilde{Q}$ of $\tilde{q}_0$ on the switching hypersurface $\tilde{\Gamma}_{i_0 i_1}$ from control i_0 to control i_1. If $\tilde{q} \in \tilde{Q}$ is sufficiently close to $\tilde{q}_0$, then the trajectory that goes through $\tilde{q}$ will intersect $\tilde{\Gamma}_{i_0 i_1}$ again in some point $\tilde{P}'(\tilde{q}) \in \tilde{Q}$ after s' control switchings. The mapping $\tilde{P}'$ that takes $\tilde{q}$ to $\tilde{P}'(\tilde{q})$ is called the Poincaré map associated with the periodic orbit ζ. The point $\tilde{q}_0$ is a fixed point of $\tilde{P}'$. It is not hard to prove that conditions (11) are sufficient for nondegeneracy of the Poincaré map at $\tilde{q}_0$.

Let us define a mapping $\tilde{P}$ on a neighbourhood of $\tilde{q}_0$. Suppose the point $\tilde{q} \in \tilde{Q}$ is sufficiently close to $\tilde{q}_0$. Consider the trajectory that goes through $\tilde{q}$. After s control switchings on this trajectory we obtain a point $\tilde{q}'$. By definition, put $\tilde{P}(\tilde{q}) = \sigma^{-1}(\tilde{q}')$. Clearly $\tilde{q}_0$ is a fixed point of the map $\tilde{P}$, and $\tilde{P}' \equiv \tilde{P}^{s'/s}$. Conditions (11) are sufficient for nondegeneracy of $\tilde{P}$ at $\tilde{q}_0$. The map $\tilde{P}$ is the Poincaré map associated with the image of ζ in orbit space Σ^*/σ.

Let us compute the maps $\tilde{P}', \tilde{P}$. Define operators $F_i(t)$ $(i = 1, \ldots, l)$ acting on T^*M by $F_i(t) : y \mapsto F(t)y + C(t)B_i$. The operator $F_i(t)$ is given by the transition matrix of the dynamical system defined by the control i.

Let $Q_k \subset T^*M$ be a neighbourhood of $q_k \in \tilde{q}_k$. Then q_k lies on the switching hypersurface $\Gamma_{i_k i_{k+1}}$ from control i_k to control i_{k+1}. Let q be a point in Q_k. By ρ_k (resp. ρ_{k+1}) denote the trajectory through q of the dynamical system on T^*M defined by control i_k (resp. i_{k+1}). The trajectories ρ_k, ρ_{k+1} intersect the hypersurface $\Gamma_{i_k i_{k+1}}$ in some points q', q''. By τ_k (resp. τ_{k+1}) denote the time that is needed to get from q to q' (resp. q'') along the trajectory ρ_k (resp. ρ_{k+1}). The functions $\tau_k(q), \tau_{k+1}(q)$ are smooth in the neighbourhood Q_k, provided Q_k is sufficiently small. These functions are zero on $\Gamma_{i_k i_{k+1}}$, specifically at q_k.

Let us define the mapping $T_{i_k i_{k+1}} = F_{i_{k+1}}(\tau_k) \circ F_{i_k}(-\tau_k)$ on Q_k. Any point on $\Gamma_{i_k i_{k+1}}$ is a fixed point of $T_{i_k i_{k+1}}$, specifically q_k. We can define the following mappings on a sufficiently small neighbourhood Q_0 of q_0: $P = \sigma^{-1} \circ \mathcal{G}_{\lambda^{-1}} \circ T_{i_s i_{s+1}} \circ F_{i_s}(t_s) \circ \cdots \circ T_{i_1 i_2} \circ F_{i_1}(t_1)$, $P' = \mathcal{G}_{(\lambda')^{-1}} \circ T_{i_{s'} i_{s'+1}} \circ F_{i_{s'}}(t_{s'}) \circ \cdots \circ T_{i_1 i_2} \circ F_{i_1}(t_1)$. The point q_0 is a fixed point of the mappings P, P'. These mappings commute with the action of $\mathcal{G}$. The mapping $F_{i_1}(\tau_1)$, which projects Q_0 on the switching surface $\Gamma_{i_0 i_1}$ along the trajectories of the system defined by control i_1, also commutes with $\mathcal{G}$. Hence the compositions $F_{i_1}(\tau_1) \circ P$, $F_{i_1}(\tau_1) \circ P'$ induce mappings of a neighbourhood of $\tilde{q}_0$ on the switching surface $\tilde{\Gamma}_{i_0 i_1}$ in Σ^*. It is not hard to see that these induced mappings coincide with the Poincaré maps $\tilde{P}$, $\tilde{P}'$. Therefore the linear part of $\tilde{P}$, $\tilde{P}'$ can be computed from the linear part of the mappings P, P' at q_0.

Note that the phase velocity vector $v_t = Ay(q_0) + b_{i_1}$ is an eigenvector of the Jacobi matrices $\frac{\partial P(y)}{\partial y}|_{y=y(q_0)}$, $\frac{\partial P'(y)}{\partial y}|_{y=y(q_0)}$ with eigenvalues $\lambda^{-1}, (\lambda')^{-1}$ respectively. Hence the differentials of P and P' induce linear operators on the quotient space $T_{q_0}(T^*M)/\mathrm{span}\{v_t\}$.

Since any point of the switching surface $\Gamma_{i_0 i_1}$ is invariant with respect to the projec-

tion operator $F_{i_1}(\tau_1)$, the differential of the restriction of the mapping $F_{i_1}(\tau_1)\circ P$ (resp. $F_{i_1}(\tau_1)\circ P'$) on $\Gamma_{i_0 i_1}$ coincides with the differential of P (resp. P'). Hence the action of the differentials $D(F_{i_1}(\tau_1)\circ P)$, $D(F_{i_1}(\tau_1)\circ P')$ on the tangent space $T_{q_0}\Gamma_{i_0 i_1}$ can be identified with the action of the differentials of P, P' on the quotient space $T_{q_0}(T^*M)/\mathrm{span}\{v_t\}$.

The vector v_λ given by $\frac{\partial}{\partial\lambda}\mathcal{G}_\lambda(y(q_0))|_{\lambda=1} = \frac{dC(\lambda)}{d\lambda}|_{\lambda=1}y(q_0)$ in the tangent space to q_0 is tangent to the orbit $\tilde{q}_0$. Since the mappings $F_{i_1}(\tau_1)\circ P$, $F_{i_1}(\tau_1)\circ P'$ commute with the action of $\mathcal{G}$, v_λ is an eigenvector of the differentials $D(F_{i_1}(\tau_1)\circ P)$, $D(F_{i_1}(\tau_1)\circ P')$ with eigenvalue 1. The vector v_λ consists of the components of the vectors $y^{2n}(q_0), 2y^{2n-1}(q_0), \ldots, 2ny^1(q_0)$ written one after the other. Hence the differentials $D(F_{i_1}(\tau_1)\circ P)$, $D(F_{i_1}(\tau_1)\circ P')$ induce an action on the quotient space $T_{q_0}\Gamma_{i_0 i_1}/\mathrm{span}\{v_\lambda\}$. This action can be identified with the action of the differentials of the Poincaré maps $\tilde{P}$, $\tilde{P}'$ on the tangent space $T_{\tilde{q}_0}\tilde{\Gamma}_{i_0 i_1}$. We proved the following assertion.

Corollary 3.1. *The action of the differentials of the Poincaré maps $\tilde{P}$, $\tilde{P}'$ can be canonically identified with the action induced by the differentials of the mappings P, P' on the quotient space $T_{q_0}(T^*M)/\mathrm{span}\{v_t, v_\lambda\}$.* □

Let us compute the derivatives $\frac{\partial P}{\partial y}$, $\frac{\partial P'}{\partial y}$. Consider the differential $DT_{i_k i_{k+1}}$ at the point q_k. Note that any vector in $T_{q_k}(T^*M)$ tangent to $\Gamma_{i_k i_{k+1}}$ is invariant with respect to $DT_{i_k i_{k+1}}$. Furthermore, $DT_{i_k i_{k+1}}$ takes the phase velocity vector $Ay(q_k)+b_{i_k}$ induced by the control i_k to the phase velocity vector $Ay(q_k)+b_{i_{k+1}}$ induced by the control i_{k+1}.

Denote the base vectors in the tangent fibration $T(T^*M)$ corresponding to differentiation with respect to y_i^j by e_i^j. Let us introduce another system of base vectors e'^j_i, where $e^1_{i_k}$ is replaced by $e'^1_{i_k} = e^1_{i_k} + e^1_{i_{k+1}}$, $e^1_{i_{k+1}}$ is replaced by $e'^1_{i_{k+1}} = Ay(q_k)+b_{i_k}$, and the other vectors e'^j_i coincide with the corresponding vectors e^j_i. Denote the image of e'^j_i under the action of $DT_{i_k i_{k+1}}$ by e''^j_i. Then $e''^1_{i_{k+1}}$ is given by $e''^1_{i_{k+1}} = Ay(q_k)+b_{i_{k+1}}$, and the other vectors e''^j_i coincide with the corresponding vectors e'^j_i.

The switching surface $\Gamma_{i_k i_{k+1}}$ is given by $y^1_{i_k}(q_k) = y^1_{i_{k+1}}(q_k)$. It is easily shown that all vectors e'^j_i, e''^j_i except $e'^1_{i_{k+1}}$, $e''^1_{i_{k+1}}$ are tangent to $\Gamma_{i_k i_{k+1}}$. By (11), the vectors $Ay(q_k)+b_{i_k}$, $Ay(q_k)+b_{i_{k+1}}$ are transversal to $\Gamma_{i_k i_{k+1}}$. Hence the matrix E' (resp. E'') consisting of the column vectors e'^j_i (resp. e''^j_i) is nonsingular. Note that the vectors b_{i_k}, $b_{i_{k+1}}$ differ only at the i_k-th and i_{k+1}-th positions. By substituting these vectors we obtain $e''^1_{i_{k+1}} - e'^1_{i_{k+1}} = (-1)^{n-1}\frac{1}{m_{i_k}}(\sum_{i=1}^l m_i)e^{2n}_{i_k} + (-1)^n\frac{1}{m_{i_{k+1}}}(\sum_{i=1}^l m_i)e^{2n}_{i_{k+1}}$. At the $((2n-1)l+i_k)$-th position of $e'^1_{i_{k+1}}$, $e''^1_{i_{k+1}}$, which corresponds to the base vector $e^1_{i_k}$, we have the term $y^2_{i_k}(q_k)$; at the $((2n-1)l+i_{k+1})$-th position, which corresponds to the base vector $e^1_{i_{k+1}}$, we have the term $y^2_{i_{k+1}}(q_k)$. The differential $DT_{i_k i_{k+1}}$ is given by $E''(E')^{-1}$. By substituting we obtain

$$DT_{i_k i_{k+1}} = I_{2nl} + \frac{(-1)^{n-1}\frac{1}{m_{i_k}}\sum_{i=1}^l m_i}{y^2_{i_k}(q_k) - y^2_{i_{k+1}}(q_k)}e_{i_k,(2n-1)l+i_k} + \frac{(-1)^n\frac{1}{m_{i_k}}\sum_{i=1}^l m_i}{y^2_{i_k}(q_k) - y^2_{i_{k+1}}(q_k)}e_{i_k,(2n-1)l+i_{k+1}}$$
$$+ \frac{(-1)^n\frac{1}{m_{i_{k+1}}}\sum_{i=1}^l m_i}{y^2_{i_k}(q_k) - y^2_{i_{k+1}}(q_k)}e_{i_{k+1},(2n-1)l+i_k} + \frac{(-1)^{n-1}\frac{1}{m_{i_{k+1}}}\sum_{i=1}^l m_i}{y^2_{i_k}(q_k) - y^2_{i_{k+1}}(q_k)}e_{i_{k+1},(2n-1)l+i_{k+1}}, \quad (12)$$

where $e_{r,s}$ is a matrix which has a 1 at position (r,s) and is elsewhere filled with zeros. Thus the matrix $DT_{i_k i_{k+1}}$ differs from I_{2nl} at four positions, which are located at rows i_k, i_{k+1}, which correspond to the base vectors $e^{2n}_{i_k}, e^{2n}_{i_{k+1}}$, and columns $(2n-1)l+i_k, (2n-1)l+i_{k+1}$, which correspond to the base vectors $e^1_{i_k}, e^1_{i_{k+1}}$.

The mappings $F_{i_k}(t_k)$ are affine and their differential is given by $F(t_k)$. Since every element of the groups $\mathcal{S}$, $\mathcal{G}$ is a linear transformation, it coincides with its differential. Thus

$$\begin{aligned} DP &= H_{\sigma^{-1}} C(\lambda^{-1}) DT_{i_s i_{s+1}} F(t_s) \dots DT_{i_1 i_2} F(t_1), \\ DP' &= C((\lambda')^{-1}) DT_{i_{s'} i_{s'+1}} F(t_{s'}) \dots DT_{i_1 i_2} F(t_1). \end{aligned} \tag{13}$$

We obtain the following algorithm for calculating the differentials of the Poincaré maps $\tilde{P}, \tilde{P}'$ of a given s'-chain ζ that is invariant with respect to a given permutation σ.

Algorithm 2: *Calculus of the linear part of the Poincaré map of a given periodic orbit.*

1. Compute the matrices DP, DP' according to (12),(13).

2. Compute the vector $v_t = Ay(q_0) + b_{i_1}$ and the vector v_λ, which consists of the components of the vectors $y^{2n}(q_0), 2y^{2n-1}(q_0), \dots, 2ny^1(q_0)$ written one after the other. The vectors v_t, v_λ span a subspace, which is invariant with respect to the transformations DP, DP'.

3. Compute the linear transformations that are induced by DP, DP' on the quotient space $T_{q_0}(T^*M)/\mathrm{span}\{v_t, v_\lambda\}$.

The space $T_{q_0}(T^*M)$ has dimension $2n(l-1)$, whereas the matrices of the differentials DP, DP' given by (13) have size $2nl \times 2nl$. The space $T_{q_0}(T^*M)$ is an invariant subspace of these matrices. Now we shall investigate the linear transformations induced by matrices (13) on the quotient space $\mathrm{span}\{e^{2n}_1, \dots, e^1_l\}/T_{q_0}(T^*M)$.

Firstly, consider how the maps P, P' act on the quotient space of the $2nl$-dimensional space parametrized by the coordinates $y^{2n}_1, \dots, y^1_l$ with respect to the subspace T^*M. This quotient space can be parametrized by the coordinates $Y^j = \sum_{i=1}^l m_i y_i^j,\ j = 1, \dots, 2n$. Since $\sum_{i=1}^l m_i u_i = 0$, the derivatives $\dot{Y}^j$ do not depend on the control. They are given by $\dot{Y}^{2n} = 0, \dot{Y}^j = Y^{j+1},\ j = 1, \dots, 2n-1$. Hence transition by time t is a linear operator given by a triangular matrix with 1's on the diagonal. The sums Y^j are invariant with respect to the action of $\mathcal{S}$, whereas $\mathcal{G}_\lambda$ multiplies Y^j by λ^{2n+1-j}. Therefore the mapping P (resp. P') acts linearly on Y^j, and this action is given by a triangular $2n \times 2n$-matrix with $\lambda^{-1}, \dots, \lambda^{-2n}$ (resp. $\lambda'^{-1}, \dots, \lambda'^{-2n}$) on the diagonal. Obviously the induced mappings coincide with their differentials. Thus we proved the following assertion.

Corollary 3.2. *The action induced by matrices (13) on the quotient space $\mathrm{span}\{e^{2n}_1, \dots, e^1_l\}/T_{q_0}(T^*M)$ is given by diagonizable operators with simple eigenvalues $\lambda^{-1}, \dots, \lambda^{-2n}$ (resp. $\lambda'^{-1}, \dots, \lambda'^{-2n}$).* □

From the spectrum of DP, DP' we can conclude on the behaviour of the optimal synthesis in a neighbourhood of optimal self-similar trajectories. By Corollary 2.5, we have

Theorem 3.3. *Suppose ζ is an optimal s'-chain with control sequence $i_1, \dots, i_{s'}$, satisfying conditions (11). Let its Poincaré map $\tilde{P}'$ be hyperbolic, and m_- be the dimension of its stable invariant manifold. Then ζ is embedded in an $(m_- + 1)$-dimensional integral submanifold*

of Σ. The preimage of this submanifold in state space is an $(m_- + 2)$-dimensional integral submanifold, which consists of optimal trajectories. These trajectories undergo chattering with periodic sequence $\overline{i_1, \dots, i_{s'}}$ of controls when approaching the origin. □

4 Contracting groups and Poincaré map

In this section we prove that the action of $\mathcal{G}$ contracts the symplectic structure on T^*M. We will deduce some properties of the linear part of the Poincaré maps associated with optimal periodic orbits in Σ^*. We depart from a generalization of the Theorem of Lyapunov-Poincaré, whose proof is purely algebraic and omitted here for space limitation reasons.

Proposition 4.1. *Suppose B is a regular complex $n \times n$ matrix and $\Lambda \neq 0$ is a complex number. Suppose W is a complex $n \times n$ matrix such that $\Lambda B = W^T B W$. Then for any complex number $\lambda \neq 0$ and any natural number $m \in \mathbb{N}$ the equations $B[\mathrm{Ker}(W - \lambda)^m] = \mathrm{Ker}(W^T - \frac{\Lambda}{\lambda})^m$, $B^T[\mathrm{Ker}(W - \lambda)^m] = \mathrm{Ker}(W^T - \frac{\Lambda}{\lambda})^m$ are satisfied.*

Corollary 4.2. *Suppose λ is an eigenvalue of W; then $\frac{\Lambda}{\lambda}$ is also an eigenvalue of W. The dimensions of their root subspaces and their proper subspaces coincide.* □

Suppose the matrices B, W and the number Λ are real, and W has only simple eigenvalues. The space $\mathbb{R}^n$ decomposes into a direct sum of minimal invariant subspaces of the operator W. A minimal invariant subspace has dimension 1 if the corresponding eigenvalue is real, and it has dimension 2, if it corresponds to a complex-conjugated pair of eigenvalues. Denote the minimal invariant subspaces of W by $V_1, \dots, V_r$. Let us put in correspondence to each subspace V_i a number λ_i. If V_i is onedimensional, then define λ_i as the corresponding real eigenvalue. If V_i is twodimensional, then define λ_i as the corresponding complex eigenvalue that has positive imaginary part. By $\bar{}$ denote complex conjugation.

Definition 4.3. We call a minimal invariant subspace V_j conjugated to the subspace V_i, if $\lambda_j \bar{\lambda}_i = \Lambda$.

By Corollary 4.2, for any subspace V_i there exists a conjugated subspace V_j. Suppose V_i, V_j are conjugated subspaces. Then we have $V_i = V_j$ if and only if $|\lambda_i|^2 = \Lambda$.

Proposition 4.4. *Suppose the assumptions made above are satisfied. Then*
a) For any minimal invariant subspace V_i of W there exist vectors $w_i \in V_i$, $w_j \in V_j$ such that $w_i^T B w_j \neq 0$. Here V_j is the subspace conjugated to V_i.
b) Suppose the minimal invariant subspaces V_i, V_j are not conjugated. Then for any vectors $w_i \in V_i$, $w_j \in V_j$ we have $w_i^T B w_j = 0$.

The proof uses the theorem on the Jordan structure of a matrix and is omitted here.

Consider a differentiable manifold V of even dimension $2m$. Let ω be a closed nondegenerate differential 2-form inducing a symplectic structure on V. By $T_z V$ denote the tangent space to V at the point $z \in V$. Then to any vector $v \in T_z V$ a 1-form $\theta_v \in T_z^* V$ in the cotangent space at the point z is assigned. The form θ_v takes any vector $u \in T_z V$ to $\theta_v(u) = \omega(v, u)$ and is the convolution $i_v \omega$ of the form ω with the vector v.

Definition 4.5. A one-parametric group $\mathbf{G}$ of diffeomorphisms g_γ of V, where $\gamma \in \mathbb{R}$ is an additive parameter, is called a *contracting group*, if the following condition holds. By the action of g_γ the form ω is multiplied by $e^{-\gamma}$, i.e. for any point $z \in V$ we have $\omega(z) = e^{-\gamma} g_\gamma^* \omega(g(z))$.

Suppose L is a Lagrange submanifold of V, i.e. a submanifold of dimension m on whose tangent space ω is zero. Then for any $z \in L$ and $v \in T_z L$ we have $T_z L \subset \operatorname{Ker} \theta_v$. Obviously any element of a contracting group takes Lagrange manifolds to Lagrange manifolds.

Suppose v_G is a smooth vector field on V. It generates a one-parametric group $\mathbf{G}$ of diffeomorphisms of the manifold V.

Proposition 4.6. *The following conditions are equivalent:*
(i) the group $\mathbf{G}$ *is a contracting group,*
(ii) the form $\theta_{v_G} = i_{v_G}\omega$ *satisfies the equation* $d\theta_{v_G} = \omega$.
Proof. By $\mathcal{L}_v$ denote the Lie derivative with respect to the vector field v. By definition, a group is contracting iff for any point $z \in V$ and any two vectors $u, w \in T_z V$ we have $\omega(g_\gamma(z))(Dg_\gamma(u), Dg_\gamma(w)) = e^\gamma \omega(z)(u, w)$. This equation is satisfied iff the equation $\mathcal{L}_{v_G}\omega = \omega$ holds. Since ω is closed, we obtain $\mathcal{L}_{v_G}\omega = d(i_{v_G}\omega) = d\theta_{v_G}$. □

Suppose $v_t(z)$ is a vector field on V. Parametrize the trajectories of the corresponding flow by time t. It is well-known that the form $\theta_{v_t}(z)$ is closed if and only if the symplectic form ω is invariant with respect to transitions along the trajectories of v_t. In this case v_t is a Hamiltonian flow. If θ_{v_t} is exact, then there exists a Hamiltonian $H(z)$ such that $\theta_{v_t} = -dH$.

Let L be a continuously differentiable integral Lagrange manifold, which is invariant with respect to the action of a contracting group $\mathbf{G}$. Then the generating vector field v_G of $\mathbf{G}$ and the vector field v_t are tangent to L. Suppose there exists a point $z_0 \in L$ and numbers $T > 0$, $\gamma \neq 0$ such that $g_\gamma \circ \Phi_T(z_0) = z_0$, where g_γ is an element of the group $\mathbf{G}$ and Φ_T is a shift by time T along the trajectories of the Hamiltonian system. The differential of the mapping $g_\gamma \circ \Phi_T$ is an automorphism of the tangent space $T_{z_0}V$. Suppose u, v are tangent vectors at the point z_0. Since the form ω is invariant under shifts in time, we have $\omega(u, v) = \omega(D\Phi_T(u), D\Phi_T(v))$, where $D\Phi_T(u), D\Phi_T(v)$ are tangent vectors at the point $\Phi_T(z_0)$. On the other hand, the action of the differential Dg_γ multiplies the form ω by e^γ. Therefore we have

$$\omega(D(g_\gamma \circ \Phi_T)(u), D(g_\gamma \circ \Phi_T)(v)) = e^\gamma \omega(u, v). \tag{14}$$

By W denote the matrix of the linear mapping $D(g_\gamma \circ \Phi_T)$, by ω_{z_0} denote the skew-symmetric matrix that corresponds to the form $\omega(z_0)$. Then (14) becomes $e^\gamma \omega_{z_0} = W^T \omega_{z_0} W$.

Since L is invariant under any shift Φ_T and any diffeomorhism g_γ, the subspace $T_{z_0}L$ of the tangent space $T_{z_0}V$ is invariant under the mapping W.

Suppose W has only simple eigenvalues. Then the space $T_{z_0}V$ decomposes into a direct sum of minimal invariant subspaces of W. Denote these subspaces by $V_1, \ldots, V_r$. The tangent space $T_{z_0}L$ is an invariant subspace of W. Hence there exists a subset $V_S \subset \{V_1, \ldots, V_r\}$ such that $T_{z_0}L = \operatorname{span}\{v \in V_i \,|\, V_i \in V_S\}$. Since $T_{z_0}L$ is isotropic, for any $V_i, V_j \subset T_{z_0}L$ and $v_i \in V_i, v_j \in V_j$ we have $\omega(v_i, v_j) = 2v_i^T \omega_{z_0} v_j = 0$.

Since the matrix ω_{z_0} is nonsingular, the matrix W satisfies the assumptions of Propositions 4.1 and 4.4. By Corollary 4.2, the set of eigenvalues of W breaks up into pairs. The product of the eigenvalues in each pair equals e^γ.

By Proposition 4.4, conjugated subspaces V_i, V_j cannot at the same time be contained in the set V_S. Since $\dim T_{z_0}L = \frac{1}{2}\dim T_{z_0}V$, there cannot exist any subspace V_i that coincides with its conjugated subspace. Hence the number of minimal invariant subspaces is even. There exist exactly $2^{\frac{r}{2}}$ Lagrange subspaces of $T_{z_0}V$ that are invariant under W.

Let us summarize these results.

Proposition 4.7. *Suppose the matrix $W = D(g_\gamma \circ \Phi_T)$ has only simple eigenvalues. Then the minimal invariant subspaces $V_1, \dots, V_{2r}$ of W and the corresponding eigenvalues $\lambda_1, \dots, \lambda_{2r}$ with nonnegative imaginary part can be arranged in a manner such that the following conditions hold.*
a) $T_{z_0}L = V_1 \oplus V_2 \oplus \dots \oplus V_r$.
b) $\lambda_i\,\bar\lambda_{r+i} = \bar\lambda_i\,\lambda_{r+i} = e^\gamma$ *for any* $i = 1, \dots, r$.
c) For any $i = 1, \dots, r$ *there exist* $v_i \in V_i, v_{r+i} \in V_{r+i}$ *such that* $\omega(v_i, v_{r+i}) \neq 0$.
d) For any $v_i \in V_i, v_j \in V_j$ *such that* $(j-i) \not\equiv 0 \bmod(r)$ *we have* $\omega(v_i, v_j) = 0$.
e) $\dim V_i = \dim V_{r+i}$ *for any* $i = 1, \dots, r$. □

Suppose the differential of the diffeomorphism g_γ multiplies the vector field v_t by $e^{\kappa\gamma}$, $\kappa > 0$, i.e. at any point $z \in V$ we have $Dg_\gamma(v_t(z)) = e^{\kappa\gamma}v_t(g_\gamma(z))$. This equation holds iff $\mathcal{L}_{v_G}v_t = -\kappa v_t$, i.e.

$$[v_t, v_G] = \kappa v_t. \tag{15}$$

Let us compute the images $W(v_t), W(v_G)$. We have $D\Phi_T(v_t(z_0)) = v_t(\Phi_T(z_0))$. Now we shall compute $D\Phi_T(v_G(z_0))$. We have $\mathcal{L}_{v_t}v_G = \kappa v_t$, therefore $v_G(\Phi_T(z_0)) = D\Phi_T(v_G(z_0)) + \kappa T v_t(\Phi_T(z_0))$. Hence we obtain $D\Phi_T(v_G(z_0)) = (v_G - \kappa T v_t)(\Phi_T(z_0))$. The differential Dg_γ multiplies the vector field v_t by $e^{\kappa\gamma}$ and leaves v_G invariant. This yields

$$W(v_t) = e^{\kappa\gamma}v_t, \qquad W(v_G) = v_G - \kappa e^{\kappa\gamma}T v_t.$$

Thus the vectors v_t and $v_G + \frac{\kappa T}{1-e^{-\kappa\gamma}}v_t$ are eigenvectors of the matrix W with eigenvalues $e^{\kappa\gamma}$ and 1, respectively.

Suppose v_t, v_G are linearly independent at z_0. Denote the $(2m-2)$-dimensional quotient space $T_{z_0}V/\mathrm{span}\{v_t, v_G\}$ by $\tilde V$. The linear operator W induces an automorphism $\tilde W$ of $\tilde V$. Since $v_t, v_G \in T_{z_0}L$, the quotient space $\tilde L = T_{z_0}L/\mathrm{span}\{v_t, v_G\}$ is well-defined. It is a $(m-2)$-dimensional subspace of $\tilde V$. By Proposition 4.7, the following assertion holds.

Proposition 4.8. *Suppose W has only simple eigenvalues, and the assumptions made above are satisfied. Then the $2r-2$ minimal subspaces $\tilde V_1, \dots, \tilde V_{2r-2}$ of $\tilde V$ that are invariant under the action of $\tilde W$, and the corresponding eigenvalues $\lambda_1, \dots, \lambda_{2r-2}$ with nonnegative imaginary part can be arranged in a manner such that the following conditions hold.*
a) $\tilde L = \tilde V_1 \oplus \tilde V_2 \oplus \dots \oplus \tilde V_{r-2}$.
b) $\lambda_i\,\bar\lambda_{r-2+i} = \bar\lambda_i\,\lambda_{r-2+i} = e^\gamma$ *for any* $i = 1, \dots, r-2$.
c) $\dim \tilde V_{2r-3} = \dim \tilde V_{2r-2} = 1, \lambda_{2r-3} = e^{(1-\kappa)\gamma}, \lambda_{2r-2} = e^\gamma$.
d) $\dim \tilde V_i = \dim \tilde V_{r-2+i}$ *for any* $i = 1, \dots, r-2$. □

Now we apply these results to the self-similar trajectories of system (4). In our case V is the phase space T^*M with its canonical symplectic structure. The section of T^*M corresponding to the optimal synthesis is a Lagrange submanifold.

Proposition 4.9. *The Fuller group $\mathcal{G}$ is a contracting group. The parameters γ and λ are related to each other by the equation $\lambda^{2n+1} = e^{\gamma}$.*

Proof. We have $v_G(q) = \frac{d\mathcal{G}_\lambda(q)}{d\lambda}(\frac{d((2n+1)\ln\lambda)}{d\lambda})^{-1}|_{\lambda=1} = \frac{1}{2n+1}\frac{dC(\lambda)}{d\lambda}y(q)$. Hence the relation between the generating vector field v_G and the vector field v_λ, which was defined in the previous section, is given by

$$v_G(q) = \frac{1}{2n+1}v_\lambda = \frac{1}{2n+1}\sum_{i=1}^{l}\sum_{j=1}^{2n}(2n+1-j)y_i^j(q)\frac{\partial}{\partial y_i^j}.$$

The phase velocity vector v_t is given by

$$v_t(q) = Ay(q) + b_k = \sum_{i=1}^{l}\sum_{j=2}^{2n} y_i^{j-1}(q)\frac{\partial}{\partial y_i^j} + \sum_{i=1}^{l} u_i\frac{\partial}{\partial y_i^1}.$$

Here k is the applied control. It is easily checked that the Lie bracket $[v_t, v_G]$ is equal to $\frac{1}{2n+1}v_t$. Therefore condition (15) with $\kappa = \frac{1}{2n+1}$ is satisfied.

The symplectic form on $V = T^*M$ is given by $\omega = \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{2n}(-1)^{n+1-j}m_i dy_i^{2n+1-j}\wedge dy_i^j$. In coordinate representation we have

$$(\omega_{ij}) = \frac{1}{2}\begin{pmatrix} 0 & 0 & \dots & 0 & (-1)^n\Delta_m \\ 0 & 0 & \dots & (-1)^{n-1}\Delta_m & 0 \\ \vdots & \vdots & \iddots & \vdots & \vdots \\ 0 & (-1)^{-n+2}\Delta_m & \dots & 0 & 0 \\ (-1)^{-n+1}\Delta_m & 0 & \dots & 0 & 0 \end{pmatrix}. \qquad (16)$$

Here $\Delta_m = \mathrm{diag}\{m_1, \dots, m_l\}$ is a diagonal $l \times l$-matrix.

Hence the form $\theta_{v_G} = i_{v_G}\omega$ is given by

$$\theta_{v_G}(q) = \frac{1}{2n+1}\sum_{i=1}^{l}\sum_{j=1}^{2n}(-1)^{n+1-j}m_i j y_i^{2n+1-j}(q)dy_i^j.$$

It is not hard to prove that the differential of this form after alternation is equal to (16). Proposition 4.6 concludes the proof. □

Since the mappings $\sigma^{-1}\circ T_{i_s i_{s+1}}\circ F_{i_s}(t_s)\circ\dots\circ T_{i_1 i_2}\circ F_{i_1}(t_1)$, $T_{i_{s'} i_{s'+1}}\circ F_{i_{s'}}(t_{s'})\circ\dots\circ T_{i_1 i_2}\circ F_{i_1}(t_1)$ are not transitions in time of system (4), Proposition 4.7 cannot be applied formally to the mappings P and P'. But recall that for proving Propositions 4.7 and 4.8 we used only that Φ_T preserves the form ω. The mappings $F_i(t)$ are transitions in time and preserve ω. Clearly the action of the group $\mathcal{S}$ preserves ω. The differential of the mapping $T_{i_k i_{k+1}}$ can be represented as a composition of differentials of the mappings $F_{i_{k+1}}(\tau_k)$ and $F_{i_k}(-\tau_k)$ with

frozen argument τ_k and the differential of $T_{i_k i_{k+1}}$ at some point on the switching surface $\Gamma_{i_k i_{k+1}}$. The first two differentials are differentials of transitions in time and hence preserve ω. By multiplying (12) and (16), we obtain $(\omega_{ij}) = (DT_{i_k i_{k+1}})^T(\omega_{ij})DT_{i_k i_{k+1}}$. Therefore $T_{i_k i_{k+1}}$ also preserves ω.

Note that on any self-similar trajectory corresponding to an s'-chain with $s' > 1$ the vector fields v_G and v_t are linearly independent. Hence Propositions 4.7 and 4.8 remain valid also for system (4). By L denote the Lagrange section of the cotangent fibration T^*M that is induced by the optimal synthesis. We proved the following assertion on the differentials of the mappings P, P'.

Theorem 4.10. *Suppose the differential DP has only simple eigenvalues. Then the minimal subspaces $V_1, \dots, V_{2r}$ of the space $T_{q_0}(T^*M)$ that are invariant under the action of DP, and the corresponding eigenvalues $\lambda_1, \dots, \lambda_{2r}$ with nonnegative imaginary part can be arranged in a manner such that the following conditions hold.*
a) If L is differentiable at q_0, then $T_{q_0}L = V_1 \oplus V_2 \oplus \dots \oplus V_r$.
b) $\lambda_i\,\bar{\lambda}_{r+i} = \bar{\lambda}_i\,\lambda_{r+i} = \lambda^{-(2n+1)}$ for any $i = 1, \dots, r$.
c) For any $i = 1, \dots, r$ there exist $v_i \in V_i, v_{r+i} \in V_{r+i}$ such that $\omega(v_i, v_{r+i}) \neq 0$.
d) For any $v_i \in V_i, v_j \in V_j$ such that $(j - i) \not\equiv 0 \bmod(r)$ we have $\omega(v_i, v_j) = 0$.
e) $\lambda_1 = \lambda^{-1},\ \lambda_2 = 1,\ V_1 = \operatorname{span}\{v_t\},\ V_2 = \operatorname{span}\{v_\lambda + \frac{\sum_{i=1}^{s} t_i}{1-\lambda} v_t\}$.
f) $\dim V_i = \dim V_{r+i}$ for any $i = 1, \dots, r$.

Here the vector v_t is the phase velocity vector defined by control i_1, and the vector v_λ is tangent to the orbit $\tilde{q}_0$.

Analogous assertions hold for the differential DP', with λ, s replaced by λ', s'. □

By $\tilde{L}$ denote the image of the intersection $L \cap \Gamma_{i_0 i_1}$ in space Σ^*. The results on the differentials of the Poincaré maps $\tilde{P}$, $\tilde{P}'$ can be summarized as follows.

Theorem 4.11. *Suppose the differential DP has only simple eigenvalues. Then the minimal subspaces $\tilde{V}_1, \dots, \tilde{V}_{2r-2}$ of the space $T_{\tilde{q}_0}\tilde{\Gamma}_{i_0 i_1}$ that are invariant under the action of the differential $D\tilde{P}$, and the corresponding eigenvalues $\lambda_1, \dots, \lambda_{2r-2}$ with nonnegative imaginary part can be arranged in a manner such that the following conditions hold.*
a) If $\tilde{L}$ is differentiable at $\tilde{q}_0$, then $T_{\tilde{q}_0}\tilde{L} = \tilde{V}_1 \oplus \tilde{V}_2 \oplus \dots \oplus \tilde{V}_{r-2}$.
b) $\lambda_i\,\bar{\lambda}_{r-2+i} = \bar{\lambda}_i\,\lambda_{r-2+i} = \lambda^{-(2n+1)}$ for any $i = 1, \dots, r-2$.
c) $\dim \tilde{V}_{2r-3} = \dim \tilde{V}_{2r-2} = 1, \lambda_{2r-3} = \lambda^{-2n}, \lambda_{2r-2} = \lambda^{-(2n+1)}$.
d) $\dim \tilde{V}_i = \dim \tilde{V}_{r-2+i}$ for any $i = 1, \dots, r-2$.

Analogous assertions hold for the differential $D\tilde{P}'$, with λ replaced by λ'. □

Note that the spectrum of the matrices (13) does not coincide with the spectrum of the differentials DP, DP'. By Corollary 3.2, the eigenvalues $\lambda^{-1}, \lambda^{-2n}$ (resp. $(\lambda')^{-1}, (\lambda')^{-2n}$) of matrices (13) are always multiple, whereas absence of multiple eigenvalues in the spectrum of the differentials DP, DP' is the generic case. Corollary 3.2 yields the following criterion of absence of multiple eigenvalues in the spectrum of the differentials DP, DP'.

Proposition 4.12. *The differential DP (resp. DP') has only simple eigenvalues if and only if the numbers $\lambda^{-1}, \dots, \lambda^{-2n}$ (resp. $\lambda'^{-1}, \dots, \lambda'^{-2n}$) are eigenvalues of the corresponding matrix (13) with multiplicity not greater than 2, and any other eigenvalue is simple.* □

5 Conclusions

When constructing an optimal synthesis in a deterministic control problem, one usually first considers singular trajectories and submanifolds provided by the maximum principle. These act as structuring elements in the phase portrait. In classical dynamical systems, however, the structuring elements are fixed points, periodic cycles and associated invariant submanifolds. If the optimal control problem possesses a certain symmetry group (a Fuller group), one can consider these classical objects in orbit space with respect to the group. Computing them yields valuable information on the global structure of optimal synthesis.

In this paper we provide tools and algorithms to compute such elements. To this end we expolited the interaction of the symplectic structure of the Hamiltonian dynamics emanating from the maximum principle on the one hand and the Fuller group on the other hand.

References

[1] Fuller A.T. Relay control systems optimized for various performance criteria. In *Proc. of the First Internat. Congr. of the IFAC, Moscow*, Vol. 1 (London, U.K.: Butterworth), 1960, 510–519.

[2] Fuller A.T. Dimensional properties of optimal and sub-optimal nonlinear control systems. *Journal of Franklin Institute* 289 (1970), 379–393.

[3] Fuller A.T. Constant-ratio trajectories in optimal control systems. *Int. J. Contr.* 58 (1993), no.6, 1409–1435.

[4] Hildebrand R. An open problem in optimal control theory. *Journal of Mathematical Sciences* 121 (2004), no.2, 2178–2220.

[5] Marchal C. Chattering arcs and chattering controls. *Journal of Optimization Theory and Applications* 11 (1973), 441–468.

[6] Pontryagin L.S., Boltyanskii V.G., Gamkrelidze R.V., Mishchenko E.F. *The mathematical theory of optimal processes.* Wiley, New York, 1962.

[7] Zelikin M.I., Borisov V.F. *Theory of Chattering Control with Applications to Astronautics, Robotics, Economics and Engineering.* Birkhäuser, Boston, 1994.

Roland Hildebrand
LMC, IMAG
Université Joseph Fourier
roland.hildebrand@imag.fr

Sudoku y Bases de Gröbner (Sudoku and Gröbner Bases)

Jorge Martín-Morales

Abstract

Sudoku is a logic-based placement puzzle. The aim of the canonical puzzle is to enter a numeral from 1 through 9 in each cell of a 9×9 grid made up of 3×3 regions, starting with various numerals given in some cells. Each row, column and region must contain only one instance of each numeral. Proper sudoku puzzle should have unique solutions. There are many variants of this game. In this paper we give a new method which can be applied to solving most of those variants using Gröbner bases techniques. Moreover, if a sudoku has more than one solution, we can also compute all of them.

Resumen

Sudoku es un rompecabezas de colocación cuyo objetivo es rellenar una cuadrícula de 9×9 celdas dividida en 9 regiones de 3×3 con las cifras del 1 al 9, partiendo de algunos números ya dispuestos en algunas celdas y de manera que no se repita ninguna cifra en una misma fila, columna o región. Un sudoku es compatible determinado si tiene solución única. Existen muchas variantes de este juego con sus correspondientes resolutores. En este trabajo presentamos un nuevo método basado en técnicas de bases de Gröbner y aplicable a muchas de esas variantes que además nos permite saber si un sudoku es compatible indeterminado y en caso afirmativo calcular todas las soluciones.

Introducción

Sudoku es un interesante pasatiempo que se popularizó en Japón en 1986 y se dio a conocer en el ámbito internacional en 2005, aunque su origen fue en New York con el nombre de "Number Place". El objetivo es rellenar una cuadrícula de 9×9 celdas dividida en 9 regiones de 3×3 con las cifras del 1 al 9, partiendo de algunos números ya dispuestos en algunas celdas y de manera que no se repita ninguna cifra en una misma fila, columna o región 3×3, (figura 1). Un sudoku es compatible determinado si la solución es única. Hay muchas variantes, siendo la más común la de cuadrícula 9×9 con regiones 3×3, pero también se utilizan otros tamaños. Además, las regiones no tienen por qué ser cuadradas. El problema general de resolver un sudoku de tamaño $n^2 \times n^2$ es NP-completo.

Para $n = 3$, un sudoku puede interpretarse como un problema de coloreado de un cierto grafo de 81 vértices, con 9 colores. Una técnica para obtener este coloreado es mediante la

resolución de un sistema polinómico. Esto es discutido en la primera sección donde damos un ejemplo de un grafo 3-coloreable en el que se muestra cómo las bases de Gröbner nos permiten calcular todas las soluciones. En la segunda sección explicamos cómo podemos interpretar un sudoku como un grafo 9-coloreable. También damos un ejemplo de un sudoku que tiene 98 soluciones e indicamos cómo calcularlas todas usando SINGULAR. Al final de la segunda sección citamos algunas variantes de sudoku que también pueden modelizarse con un sistema de ecuaciones polinómicas. En la tercera sección dejamos algunas preguntas abiertas.

	9				4			7
					7	9		
8								
4		5	8					
3								2
					9	7		6
								4
		3	5					
2			6				8	

Figura 1: sudoku con 20 datos

Las bases de Gröbner juegan un papel esencial en Geometría Algebraica y Álgebra Conmutativa. Sin embargo tienen algunas aplicaciones mucho más básicas y que alguno de nuestros alumnos desconocen, entre ellas resolver un sistema de ecuaciones polinómicas. Pensamos que este papel tiene un alto contenido didáctico y podría usarse por los profesores para mostrar la importancia de las bases de Gröbner a la hora de resolver sistemas de ecuaciones polinómicas no lineales. Al mismo tiempo se puede aprovechar para motivar el aprendizaje de algún programa de cálculo simbólico como puede ser SINGULAR, Macaulay2 o CoCoA. Todos los cálculos realizados en este trabajo se hicieron en una máquina AMD Opteron 252–64 bit, 2,6Ghz–RAM 8 Gb en SINGULAR–3–0–0, bajo el sistema operativo Fedora Core 4.

1. El problema del n-coloreado de un grafo

El problema del coloreado de un grafo puede resolverse mediante el cálculo de las soluciones de un sistema algebraico [1]. En esta sección incluimos una parte de la discusión de [1], con algunos comentarios y explicamos cómo podemos resolver ejemplos concretos con la ayuda de un programa de cálculo simbólico, (SINGULAR [2, 3]).

El problema que se plantea es el siguiente. Dado un grafo $\mathcal{G}$ con m vértices y a lo más una arista entre dos vértices, queremos saber si es posible colorear los vértices usando n colores de tal manera que dos vértices adyacentes tengan distinto color.

Consideramos $\omega = e^{\frac{2\pi i}{n}}$ una raíz primitiva n-ésima de la unidad y representamos los n colores por $1, \omega, \ldots, \omega^{n-1}$, las n raíces de $x^n - 1$. Sean ahora $x_1, \ldots, x_m$ las variables

que representan los m vértices de $\mathcal{G}$. Vamos a intentar encontrar unas ecuaciones para las variables x_i. Cada vértice x_i tiene asignado uno de los n colores. Esto puede representarse por las ecuaciones

$$x_i^n - 1 = 0, \quad 1 \leq i \leq m. \tag{1}$$

También si dos vértices x_i y x_j son adyacentes entonces tienen que tener distinto color. Puesto que $x_i^n = x_j^n$, entonces $(x_i - x_j)(x_i^{n-1} + x_i^{n-2}x_j + \cdots + x_i x_j^{n-2} + x_j^{n-1}) = 0$. Nótese que el factor $x_i - x_j$ aparece una sola vez en el polinomio $x_i^n - x_j^n$. Luego x_i y x_j tienen distinto color si y sólo si

$$\sum_{k+l=n-1} x_i^k x_j^l = 0. \tag{2}$$

Ya tenemos el sistema de ecuaciones buscado. Sea I el ideal de $\mathbb{C}[x_1, \ldots, x_m]$ generado por los polinomios de la ecuación (1) y los de la ecuación (2) para los pares (i, j) tales que x_i y x_j sean adyacentes. Entonces el grafo $\mathcal{G}$ es n-coloreable si y sólo si la variedad $V(I) \neq \emptyset$. Teniendo en cuenta el Nullstellensatz de Hilbert [1], el siguiente resultado es claro.

Teorema 1.1. *El grafo $\mathcal{G}$ es n-coloreable si y sólo si el ideal I no es el total.*

Una forma de proceder es calcular una base de Gröbner G del ideal I y preguntarnos si G contiene una constante. En caso negativo el grafo $\mathcal{G}$ será n-coloreable y la base de Gröbner nos da mucha información acerca del coloreado del grafo. A partir de ella podemos calcular una solución, y si tiene más de una, podemos saber cuántas tiene e incluso calcularlas todas.

Nota 1.2. Para calcular una base de Gröbner de I, dado que todos los polinomios de I están en $\mathbb{Q}[\mathbf{x}]$, podemos calcular una base de Gröbner en $\mathbb{Q}[\mathbf{x}]$ y lo que obtengamos también será una base de Gröbner en $\mathbb{C}[\mathbf{x}]$. Ahora bien, si queremos calcular una solución particular entonces tenemos que introducir un polinomio de la forma $x_i - \omega^a$ y ahora $I \nsubseteq \mathbb{Q}[\mathbf{x}]$.

Nota 1.3. Por las condiciones del problema existe un número finito de soluciones, así que el ideal I es 0-dimensional y $\dim_{\mathbb{C}} \left(\frac{\mathbb{C}[\mathbf{x}]}{I}\right) = \#\mathbb{N}^m \setminus E(I)$ es el número de soluciones, donde $E(I) = \{exp(f) \mid f \in I \setminus \{0\}\}$ denota la escalera del ideal respecto de un orden monomial.

Ejemplo 1.4. Consideramos el grafo $\mathcal{G}$ de la figura 2

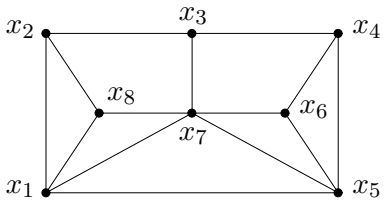


Figura 2: grafo 3-coloreable

El ideal I está formado por los polinomios $x_i^3 - 1$, para $i = 1, \ldots, 8$ y por $x_i^2 + x_i x_j + x_j^2$ para los pares $(i, j) \in \{(1,2), (1,5), (1,7), (1,8), (2,3), (2,8), (3,4), (3,7), (4,5), (4,6), (5,6), (5,7), (6,7), (7,8)\}$. Calculamos G una base de Gröbner de I respecto al orden graduado lexicográfico inverso (`dp`). Para ello podemos escribir en SINGULAR lo siguiente:

```
ring R = 0,x(1..8),dp;  ideal I1,I2,I,G;
for (int i=1; i<=8; i++) { I1[i] = x(i)^3-1; };
proc p (int i, int j) { return(x(i)^2+x(i)*x(j)+x(j)^2); };
I2 =  p(1,2), p(1,5), p(1,7), p(1,8), p(2,3), p(2,8), p(3,4),
      p(3,7), p(4,5), p(4,6), p(5,6), p(5,7), p(6,7), p(7,8);
I = I1 + I2;  G = std(I);
```

Obtenemos $G = \{x_6 + x_7 + x_8,\ x_5 - x_8,\ x_4 - x_7,\ x_2 - x_7,\ x_1 + x_7 + x_8,\ x_7^2 + x_7x_8 + x_8^2,\ x_3^2 + x_3x_7 - x_7x_8 - x_8^2,\ x_8^3 - 1\}$. Como $1 \notin G$, el teorema (1.1) nos dice que el grafo $\mathcal{G}$ es 3-coloreable. El número de soluciones del sistema es el número de puntos que hay por debajo de la escalera del ideal, es decir, el cardinal del conjunto $\mathbb{N}^8 \setminus E(I)$, (ver nota 1.3). Escribimos en la sesión anterior de SINGULAR `vdim(G)` y devuelve `12`. Esto quiere decir que podemos colorear el grafo $\mathcal{G}$ de 12 formas distintas, pero esencialmente distintas sólo $\frac{12}{3!} = 2$, pues el resto son permutaciones de éstas. Para calcularlas podemos hacerlo resolviendo el sistema G, que es muy sencillo. Las soluciones, salvo permutaciones, son $(1, \omega, 1, \omega, \omega^2, 1, \omega, \omega^2)$ y $(1, \omega, \omega^2, \omega, \omega^2, 1, \omega, \omega^2)$. Si a este grafo le añadimos la arista que une los vértices x_4 y x_7, vemos que el ideal $I = (1)$ y por tanto el grafo $\mathcal{G}$ con la nueva arista no es 3-coloreable.

```
I=I,p(4,7); std(I);
//-> _[1]=1
```

Nota 1.5. Supongamos que los colores son $a_1, \ldots, a_n$, números complejos distintos cualesquieras. Entonces cada vértice x_i debe cumplir la ecuación $F(x) = \prod_{k=1}^{n}(x - a_k) = 0$. Si dos vértices x_i y x_j son adyacentes entonces $0 = F(x_i) - F(x_j) = (x_i - x_j)G(x_i, x_j)$.[1] Así que dos vértices son adyacentes si y sólo si $G(x_i, x_j) = 0$. El ideal asociado al grafo $\mathcal{G}$ es el generado por los polinomios $F(x_i)$ para $i = 1, \ldots, m$ y por $G(x_i, x_j)$ para los pares (i, j) tales que x_i y x_j son adyacentes. Hemos reescrito las ecuaciones de manera que podemos representar los colores con números complejos cualesquieras. En el ejemplo (1.4) si tomamos $(a_1, a_2, a_3) = (1, 2, 3)$, entonces $G(x, y) = x^2 + xy + y^2 - 6x - 6y + 11$ en lugar de $x^2 + xy + y^2$.

2. Sudoku como un grafo 9-coloreable

El objetivo de un sudoku es rellenar una cuadrícula de 9×9 celdas dividida en 9 regiones de 3×3 con las cifras del 1 al 9, partiendo de algunos números ya dispuestos en algunas celdas y de manera que no se repita ninguna cifra en una misma fila, columna o región 3×3. Un sudoku es compatible determinado si tiene solución única. En esta sección explicamos como puede interpretarse un sudoku como un problema de coloreado parcial de un grafo y entonces, usando las técnicas de la sección 1, podemos resolver un sudoku calculando una base de Gröbner. Además si el sudoku es compatible indeterminado podemos saber cuantas soluciones tiene y calcularlas todas. También damos un ejemplo y explicamos cómo usar SINGULAR para resolverlo. Al final de la sección citamos algunas variates de sudoku.

En primer lugar enumeramos las celdas de un sudoku de la siguiente forma:

[1] Si $F(x)$ es un polinomio, $F(x) - F(y)$ tiene siempre el factor $x - y$ con multiplicidad 1 y entonces existe un polinomio $G(x, y)$ tal que $F(x, y) = (x - y)G(x, y)$, donde $G(x, y)$ no es divisible por $x - y$.

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81

Figura 3: enumeración de las celdas

Consideramos el grafo $\mathcal{G}$ formado por los vértices x_i con $i = 1, \ldots, 81$, de manera que dos vértices x_i y x_j están unidos por una arista si y sólo si los índices i y j están en la misma fila, columna o región 3×3. Así, por ejemplo, el conjunto de vértices adyacentes a x_{34} es

$$\{x_7, x_{16}, x_{25}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{33}, x_{35}, x_{36}, x_{43}, x_{44}, x_{45}, x_{52}, x_{53}, x_{54}, x_{61}, x_{70}, x_{79}\}.$$

Puesto que se trata de un grafo regular de valencia 20, el número de aristas que tiene $\mathcal{G}$ es $\frac{81 \cdot 20}{2} = 810$. Las cifras del 1 al 9 representan nueve colores. Si una celda está ocupada por un número k esto quiere decir que el vértice asociado tiene el color k. Luego resolver un sudoku es equivalente a colorear el grafo $\mathcal{G}$ con 9 colores con la condición que algunos vértices tienen que tener un color predeterminado (el indicado por los números dados). Sea I el ideal de $\mathbb{C}[\mathbf{x}]$ generado por los polinomios $F(x_i)$ con $i = 1, \ldots, 81$ y por $G(x_i, x_j)$ para los pares (i, j) tales que x_i y x_j son adyacentes, donde

$$F(x) = \prod_{i=1}^{9}(x - i), \quad G(x, y) = \frac{F(x) - F(y)}{x - y} = x^8 + x^7 y + \cdots + 1026576.$$

Si, por ejemplo, queremos resolver el sudoku de la figura 1, tenemos que añadir al ideal I los polinomios

$$x_2 - 9,\ x_6 - 4,\ x_9 - 7,\ x_{15} - 7,\ x_{16} - 9,\ x_{19} - 8,\ x_{28} - 4,\ x_{30} - 5,\ x_{31} - 8,\ x_{37} - 3,$$
$$x_{45} - 2,\ x_{51} - 9,\ x_{52} - 7,\ x_{56} - 6,\ x_{63} - 4,\ x_{66} - 3,\ x_{67} - 5,\ x_{73} - 2,\ x_{76} - 6,\ x_{80} - 8.$$

Ahora bien, una vez que hemos añadido estos polinomios se puede comprobar que las ecuaciones $F(x_i) = 0$ son redundantes y por tanto podemos quitarlas. El sistema de ecuaciones que resulta tiene 810 ecuaciones (el número de aristas de $\mathcal{G}$) más el número de datos (20 en este caso). Si el sudoku es compatible determinado, el sistema tiene solución única y la base de Gröbner reducida es de la forma $G = \{x_i - a_i \mid i = 1, \ldots, 81\}$ donde todos los a_i son números del 1 al 9. Precisamente esos a_i forman la solución buscada.

Ejemplo 2.1. Consideremos el sudoku de la siguiente figura 4.

Hemos escrito las 810 ecuaciones y una subrutina en un archivo "sudoku" que está disponibles en Internet[2]. Funciona de la siguiente forma:

```
<"sudoku";
intmat A[9][9] =
        9,0,0,0,0,0,0,0,8, 5,0,0,2,0,8,0,6,0, 0,0,3,7,1,0,0,0,9,
        0,0,0,0,7,3,0,5,0, 2,0,0,0,0,0,0,0,4, 0,5,0,1,6,0,0,0,0,
        8,0,0,0,2,7,3,0,0, 0,4,0,3,0,9,0,0,1, 7,0,0,0,0,0,0,0,2;
def G = sudoku(A); vdim(G);
//used time: 1.65 sec
//-> 1
```

El sudoku es compatible determinado y la base de Gröbner reducida G nos da la única solución.

9								8
5			2		8		6	
		3	7	1				9
				7	3		5	
2								4
	5		1	6				
8				2	7	3		
	4		3		9			1
7								2

Figura 4: sudoku con 28 datos

9	2	6	5	3	4	7	1	**8**
5	7	1	**2**	9	**8**	4	**6**	3
4	8	**3**	**7**	**1**	6	5	2	**9**
1	9	8	4	**7**	**3**	2	**5**	6
2	6	7	9	8	5	1	3	**4**
3	**5**	4	**1**	**6**	2	9	8	7
8	1	9	6	**2**	**7**	**3**	4	5
6	**4**	2	**3**	5	**9**	8	7	**1**
7	3	5	8	4	1	6	9	**2**

Figura 5: solución ejemplo 2.1

Supongamos ahora que tenemos el mismo sudoku pero las celdas 64 y 82 están vacías. Posiblemente el sudoku ya no sea compatible determinado.

```
A[6,4]=0; A[8,2]=0;
G=sudoku(A); vdim(G);
//used time: 127.71 sec
//-> 53
```

A la vista de los resultados, hay 53 soluciones distintas. Para calcularlas todas

```
LIB "solve.lib";
def S = solve(G,5,0,"nodisplay");
setring S; size(SOL);
//-> 53
SOL[1];  //Primera solucion de la lista
```

[2] http://www.us.es/gmcedm/

Siguiendo un proceso análogo podemos ver qué ocurre si la celda 26 está también vacía. En este caso existen 98 sudoku y tardó `1611,67` segundos en calcular la base reducida. También puede comprobarse que el ejemplo 2.1 con la cifra 4 en la celda 1 no tiene solución, lo cual no es inmediato a simple vista.

Nota 2.2. Para representar los 9 colores de un sudoku hemos usado los números del 1 al 9 pero, al igual que hicimos con el ejemplo 1.4, podemos usar las raíces de la unidad. Sin embargo no hemos observado una clara diferencia desde el punto de vista computacional.

2.1. Algunas variantes de sudoku

Aunque la cuadrícula más común es de 9×9 celdas con regiones 3×3, existen muchas variantes de este juego. Pueden ser de otro tamaño, 16×16 ó incluso 25×25, además las regiones no tienen por qué ser cuadradas. A veces también se exige que en las diagonales aparezcan los números del 1 al 9 y sin repetirse. El llamado sudoku samurai es otra variante de este juego. Este rompecabezas está compuesto por cinco sudokus unidos entre si, uno de ellos en el centro y los otros cuatro en las esquinas, formando una **X**.

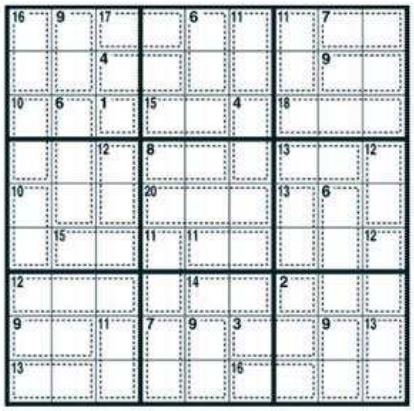


Figura 6: sudoku killer

Otra interesante variante aparece en el diario londinense The Times con el nombre de "killer sudoku". Las reglas son las mismas pero en vez de colocarse algunos números iniciales a partir de los cuales se deducen los restantes, se agrupan algunas casillas por medio de una línea punteada y se da la suma de los números agrupados, (figura 6). Para más información puede consultarse `http://www.maa.org/editorial/mathgames/mathgames_09_05_05.html`.

Todas estas variantes pueden modelizarse de manera algebraica obteniendo un sistema de ecuaciones polinómicas. Si entra en juego la aritmética de los números enteros entonces debemos representar los colores con las cifras del 1 al 9. Este es el caso del sudoku killer. En la figura 6 la casilla 1 y la 10 tienen que sumar 16. No podemos expresar esta condición si usamos las raíces de la unidad, sin embargo con los números del 1 al 9 esta condición se expresa mediante la ecuación $x_1 + x_{10} = 16$.

3. Otras cuestiones relacionadas

Es conocido que existen 6670903752021072936960 sudoku distintos[3]. Las bases de Gröbner nos proporcionan un método para saber cuantos sudoku distintos hay, pues basta con tomar como dato de entrada el sudoku vacío. Lo que ocurre es que la máquina más potente que actualmente tenemos no pudo terminar el cálculo. Planteamos algunas preguntas que están todavía abiertas:

- Se cree que 17 es el mínimo número de datos para que un sudoku sea compatible determinado[4]. Todavía no se ha encontrado ningún sudoku compatible determinado con 16 datos pero no existe ninguna prueba que diga que esto no es posible.
- Podemos modelizar un sudoku de manera algebraica y obtenemos un sistema de ecuaciones. Nos preguntamos si es posible resolverlo usando métodos numéricos con convergencia global.

Bibliografía

[1] William W. Adams and Philippe Loustaunau. *An introduction to Gröbner bases*, volumen 3 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1994.

[2] G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern (2005). `http://www.singular.uni-kl.de`. Contact the Singular team (singular@mathematik.uni-kl.de) for comments or suggestions.

[3] Gert-Martin Greuel and Gerhard Pfister. *A* **Singular** *introduction to commutative algebra*, Springer-Verlag, Berlin, 2002. With contributions by Olaf Bachmann, Christoph Lossen and Hans Schönemann, With 1 CD-ROM (Windows, Macintosh, and UNIX).

Jorge Martín Morales
Laboratorio Departamento de Álgebra
Universidad de Sevilla
Correo electrónico jormarmor@alum.us.es

[3] `http://www.shef.ac.uk/~pm1afj/sudoku/`
[4] `http://www.csse.uwa.edu.au/~gordon/sudokumin.php`

Choosing spline spaces for interpolation

Marie-Laurence Mazure

Abstract

It is well-known that Hermite interpolation by polynomial splines is possible if and only if the interpolation points satisfy classical conditions known as the Schoenberg-Whitney conditions. We recently proved that, in the most general context of splines with sections in different spaces, and with connection matrices between consecutive sections, the possibility of doing Hermite interpolation under the Schoenberg-Whitney conditions is essentially connected with existence of blossoms. We illustrate this result with the case of four-dimensional sections, taking advantage of the fact that, in this special case, we know necessary and sufficient conditions for existence of blossoms.

Introduction

In [14] Schoenberg and Whitney proved that Lagrange interpolation by polynomial splines was possible if and only if the interpolation points satisfied some conditions, known as the Schoenberg-Whitney conditions, which we shall recall in Section 1. Roughly speaking they mean that the interpolation points and the knots of the spline space must interlace in some sense. Later on, this result was extended to Hermite interpolation [4] and to other types of spline spaces (e.g., Chebsyhevian splines [15], LB-splines [7]). Many proofs concerning the Schoenberg-Whitney conditions can be found in the literature, even concerning polynomial splines. They all strongly rely on the existence of splines of minimal support (B-splines).

We recently addressed the problem of Hermite interpolation in the most general context of splines with sections in different spaces and with connection matrices between consecutive sections. In such spaces, existence of minimally supported splines is generally guaranteed by requiring total positivity for the connection matrices (see [3],[1]). When the sections are polynomial the assumption of total positivity concerns the connection matrices expressed via the ordinary derivatives. In the general case, the ordinary derivatives are to be replaced by special differential operators associated with the different sections. However total positivity is only a sufficient condition for existence of bases of B-splines, far from being necessary as we pointed out in [8]. On the other hand, in [10] we established that existence of B-splines was equivalent to existence of blossoms (see also [11]). This made it natural for us to wonder whether or not existence of blossoms was also equivalent to the Schoenberg-Whitney conditions being necessary and sufficient for Hermite interpolation. In [5] we give a positive answer to this question.

As already mentioned, existence of blossoms is a much weaker condition than total positivity of all connection matrices. Unfortunately, in the most general case of spline

spaces described above, proving existence of blossoms is far from being an easy problem to deal with. Still, in the "simple" case of splines with four dimensional sections, we were able to establish a necessary and sufficient condition for existence of blossoms, based on geometric characteristics of the sections (see [8],[9]). The latter characterisation is recalled in Section 3 where we also take advantage of it to illustrate by a few examples the validity of the Schoenberg-Whitney conditions. This also shows possible choices of spline spaces for interpolation with shape effects.

1 The Schoenberg-Whitney conditions

Given $r \geq 1$ points $x_1 < x_2 < \cdots < x_{r-1} < x_r$ in an interval I, given positive integers $\mu_1, \ldots, \mu_r$ such that $\sum_{i=1}^{r} \mu_i = N$, and given any real numbers $\alpha_{i,j}$, $1 \leq i \leq r$, $0 \leq j \leq \mu_i - 1$, consider the associated Hermite interpolation problem in N data (based on $x_1, \ldots, x_r$):

$$\text{Find } F \in \mathbb{E} \text{ such that } F^{(j)}(x_i) = \alpha_{i,j} \ , \quad 1 \leq i \leq r, \ 0 \leq j \leq \mu_i - 1, \tag{1}$$

where $\mathbb{E}$ is an N-dimensional space of sufficiently differentiable functions on I. This includes Taylor interpolation ($r = 1$, $\mu_1 = N$) as well as Lagrange interpolation ($r = N$, $\mu_1 = \cdots = \mu_N = 1$).

As is well-known, there is a unique solution to any such problem when $\mathbb{E}$ is the polynomial space $\mathbb{P}_{N-1}$ of degree less than or equal to $N-1$. For large values of N it is obviously preferable to replace the space $\mathbb{P}_{N-1}$ by an N-dimensional space of polynomial splines, defined as follows.

Choose $t_0 \leq x_1$ and $t_{q+1} \geq x_r$, a sequence $t_1 < \cdots < t_q$ of interior knots in $]t_0, t_{q+1}[$, and an associated sequence of multiplicities $m_1, \ldots, m_q$, with $1 \leq m_k \leq n$ for $1 \leq k \leq q$. Set $m := \sum_{k=1}^{q} m_k$. If x is a real number and p is a positive integer, the notation $x^{[p]}$ stands for x repeated p times. With this convention, consider the knot vector

$$\mathcal{K} := ({t_1}^{[m_1]}, \ldots, {t_q}^{[m_q]}) =: (\xi_1, \ldots, \xi_m), \text{ with } \xi_i \leq \xi_{i+1} \text{ for } 1 \leq i \leq m-1. \tag{2}$$

Associated with the knot vector $\mathcal{K}$, the polynomial spline space $\mathbb{S}$ is composed of all functions $S : [t_0, t_{q+1}] \to \mathbb{R}$ such that

(1) S is C^{n-m_k} at t_k, $1 \leq k \leq q$;

(2) for $0 \leq k \leq q$, there exists $F_k \in \mathbb{P}_n$ such that $S(t) = F_k(t)$ for all $t \in [t_k, t_{k+1}]$.

The space $\mathbb{S}$ being $(n+1+m)$-dimensional, we assume that the degree n, the number q of interior knots and the multiplicities at the interior knots are selected so that $n+1+m = N$. From now on we choose $\mathbb{E} := \mathbb{S}$.

With the notation introduced above, the sequence of interpolation points involved in problem (1) is denoted as follows

$$\mathcal{I} = ({x_1}^{[\mu_1]}, \ldots, {x_r}^{[\mu_r]}) =: (y_{-n}, \ldots, y_m), \text{ with } y_j \leq y_{j+1} \text{ for } -n \leq j \leq m-1. \tag{3}$$

To make the problem meaningful, we must require the following condition to be fulfilled

$$\text{if } x_i = t_k \text{ for some } k, \ 1 \leq k \leq q, \text{ then } \mu_i - 1 \leq n - m_k. \tag{4}$$

Nonetheless, it is well-known that the problem (1) may not be unisolvent. To obtain a unique solution it is necessary and sufficient to choose the degree, knots and multiplicities according to the Schoenberg-Whitney conditions recalled below.

Theorem 1.1. *For $1 \leq k \leq q$, we denote by $L(k)$ (resp. $R(k)$) the number of indices j, $-n \leq j \leq m$, such that $y_j < t_k$ (resp. $y_j > t_k$). We assume condition* (4) *to hold. Then, the Hermite interpolation problem* (1) *has a unique solution in the polynomial spline space* $\mathbb{S}$ *described above if and only if*

$$L(k) \geq \sum_{\ell=1}^{k} m_\ell \ , \quad R(k) \geq \sum_{\ell=k}^{q} m_\ell, \quad 1 \leq k \leq q. \tag{5}$$

Conditions (5) are known as the *Schoenberg-Whitney conditions* relative to the knot vector $\mathcal{K}$. It can be easily checked that they are equivalent to the following interlacing property linking the interpolation point sequence $\mathcal{I}$ and the knot vector $\mathcal{K}$, as introduced in (2) and (3):

$$y_{j-n-1} < \xi_j < y_j \quad \text{for } 1 \leq j \leq m. \tag{6}$$

Example 1.2. 1) In order to illustrate the latter conditions (5) or (6), consider first the case of Lagrange interpolation. Then, the Schoenberg-Whitney conditions (5) are satisfied if we choose for instance degree $n = 3$, and the interpolating points $x_3, \ldots, x_{N-2}$ as the simple (interior) knots of the polynomial spline space $\mathbb{S}$ on $[t_0, t_{q+1}] = [x_1, x_N]$ to interpolate in (cubic splines). The fact that neither x_2 nor x_{N-1} is taken as a knot of the spline space $\mathbb{S}$ is classically referred to as a *not-a-knot* procedure. Indeed, it amounts to interpolating in the $(N+2)$-dimensional space $\mathbb{S}_1$ of C^2 polynomial splines of degree 3 on $[x_1, x_N]$ with all interpolation points $x_2, \ldots, x_{N-1}$ as simple knots, imposing two additional conditions, namely $S'''(x_2^+) = S'''(x_2^-)$ and $S'''(x_{N-1}^+) = S'''(x_{N-1}^-)$, which means nothing but the fact that x_2 and x_{N-1} are no longer knots.
2) Suppose now that the interpolation problem (1) involves the values of the function and of its first derivative everywhere, that is, if $\mu_1 = \cdots = \mu_r = 2$ (hence $N = 2r$), the Schoenberg-Whitney conditions are, of course, satisfied by choosing the trivial example $n = 3$ and the interpolating points $x_2, \ldots, x_{r-1}$ as the double knots of the spline space $\mathbb{S}$. For a more sophisticated example generalising the not-a-knot procedure exactly as above, we can take $n = 7$ and double knots for the spline space $\mathbb{S}$ at the interpolating points $x_3, \ldots, x_{r-2}$. It is also possible, for instance, to define $\mathbb{S}$ with sections of degree 5, simple knots at x_2 and x_{r-1}, and double knots at $x_3, \ldots, x_{r-2}$.

2 Interpolating with shape effects

It may be interesting to interpolate with shape effects, that is, to have at our disposal one or several shape parameters to slightly modify the interpolating function or curve. This can be useful, for instance, to make up for the flaws of polynomial spline interpolation which do not always fulfil satisfying shape preservation properties. With shape effects in view,

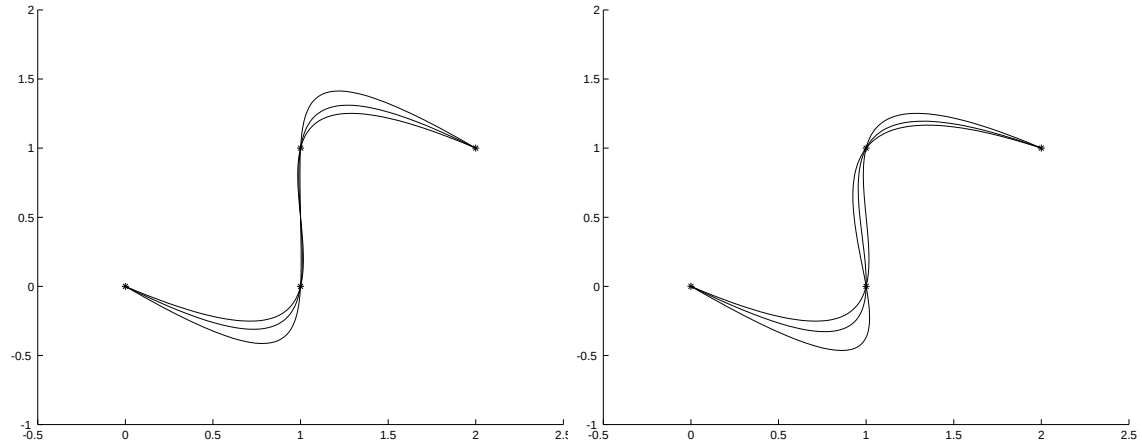


Figure 1: Regularly spaced Lagrange interpolation by parametric curves with shape effects in the space spanned by $1, x, (1-x)^p, x^q$ on $[0,1]$, with, from top to bottom on the left part of the curve, (left) $p = q = 3$ (polynomial space $\mathbb{P}_3$); $7; 11$; and (right) $q = 3$, $p = 3; 5; 8$.

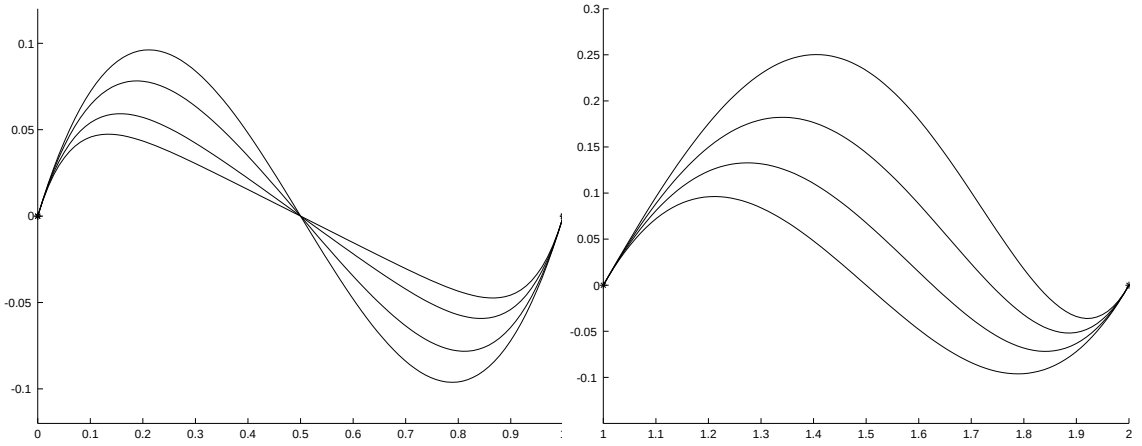


Figure 2: Hermite interpolation. Left: $F(0) = F(1) = 0$, $F'(0) = F'(1) = 1$ in the space spanned by $1, x, (1-x)^p, x^q$, with, from the most to the least oscillatory, $p = q = 3, 7; 11; 15$. Right: $F(1) = F(2) = 0$, $F'(1) = F'(2) = 1$ in the space spanned by $1, x^{k+1}, x^{k+2}, x^{k+3}$, with, from bottom to top $k = 0$ (polynomial space $\mathbb{P}_3$); $1; 2; 3$.

it is necessary to leave the framework of polynomial splines described above. The shape parameters can be of two types. Either they concern the joint at one (or several) knot(s) t_k, being then obtained by replacing the C^{n-m_k}-continuity at t_k by a continuity condition of order $n-m_k$ expressed via a connection matrix (e.g., geometrically continuous polynomial splines [3]). Or they concern the sections themselves, being then provided by the space(s) by which we replace the polynomial space of degree n (e.g., tension splines [6], variable degree polynomial splines [2]).

Figures 1 and 2 illustrate the latter type of shape parameters. In the first three cases, we interpolate in the four-dimensional space spanned by the functions $1, x, (1-x)^p, x^q$ on $[0,1]$, the shape parameters being the two integers $p, q \geq 3$. In the fourth case, we work in the four-dimensional space spanned by the functions $1, x^{k+1}, x^{k+2}, x^{k+3}$ on $]0, +\infty[$, the shape parameter k attached to this space being a non-negative integer. For the sake of completeness let us mention that all results concerning the latter spaces and their associated spline spaces are valid with less restrictive conditions on the shape parameters which are not necessarily integers, with the only requirement $p, q > 2$ and $k \geq 0$, respectively.

In spline spaces, both types of shape parameters can be cumulated : we can replace in each section the polynomial space by a space in which Hermite interpolation is possible on the corresponding interval, and at the same time we can allow connection matrices at the knots. The precise framework is as follows. Starting from the same sequence $t_0 < t_1 < \cdots < t_q < t_{q+1}$ and from the knot vector $\mathcal{K}$ introduced in (2), we suppose that

(a) for $k = 0, \ldots, q$, $\mathbb{E}_k \subset C^{n-1}([t_k, t_{k+1}])$ is an n-dimensional Extended Chebyshev space on $[t_k, t_{k+1}]$, i.e., any Hermite interpolation problem in n data has a unique solution in $\mathbb{E}_k$, or any non-zero element of $\mathbb{E}_k$ vanishes at most $n-1$ times in $[t_k, t_{k+1}]$, counting multiplicities up to n;

(b) for $k = 1, \ldots, q$, M_k is a lower triangular matrix of order $(n - m_k)$, with positive diagonal.

We now define the spline space $\mathbb{S}$ of dimension $(n+m)$ as composed of all functions $S : \cup_{k=0}^{q} [t_k^+, t_{k+1}^-] \to \mathbb{R}$ such that

(1) for $1 \leq k \leq q$, there exists $F_k \in \mathbb{E}_k$ such that $S(t) = F_k(t)$ for all $t \in [t_k^+, t_{k+1}^-]$;

(2) for $0 \leq k \leq q$, S satisfies the connection condition

$$\big(S(t_k^+), S'(t_k^+), \ldots, S^{(n-m_k-1)}(t_k^+)\big)^T = M_k \,.\big(S(t_k^-), S'(t_k^-), \ldots, S^{(n-m_k-1)}(t_k^-)\big)^T. \qquad (7)$$

Note that the presence of connection matrices in the definition of $\mathbb{S}$ implies that a spline $S \in \mathbb{S}$ is defined only separately on each $[t_k^+, t_{k+1}^-]$: in other words, for $1 \leq k \leq q$, while $S(t_k^+)$ and $S(t_k^+)$ are both meaningful, a priori $S(t_k)$ is not. Equality between two splines in $\mathbb{S}$ thus means ordinary equality on $]t_k, t_{k+1}[$ for $0 \leq k \leq q$, and equality on both t_k^- and t_k^+ for $1 \leq k \leq q$. In particular, this makes it necessary to slightly modify the way we state an Hermite interpolation problem (1) in the spline space $\mathbb{S}$ as follows:

$$\text{Find } S \in \mathbb{S} \text{ such that } S^{(j)}(x_i{}^{\varepsilon_i}) = \alpha_{i,j}\ , \quad 1 \leq i \leq r,\ 0 \leq j \leq \mu_i - 1, \qquad (8)$$

where $\varepsilon_i \in \{-, +\}$ for $1 \leq i \leq r$, and where $N = \sum_{i=1}^{r} \mu_i = n + m$. Of course if the interpolation point x_i is not a knot we can suppress ε_i. On the other hand, in any interpolation

problem we always implicitly assume the analogue of condition (4) to be fufilled, which in our present spline space $\mathbb{S}$ becomes

$$\text{if } x_i = t_k \text{ for some } k,\ 1 \le k \le q,\ \text{then } \mu_i \le n - m_k. \tag{9}$$

The result obtained in [5] is as follows.

Theorem 2.1. *The following two properties are equivalent:*

(i) *in the spline space $\mathbb{S}$ as well as in any restriction of $\mathbb{S}$ to a subinterval, as well as in any spline space (with n-dimensional section spaces) containing $\mathbb{S}$, any Hermite interpolation problem has a unique solution if and only if the corresponding Schoenberg-Whitney conditions are satisfied;*

(ii) *blossoms exist in the spline space $\widehat{\mathbb{S}}$ obtained from $\mathbb{S}$ by integration, namely*

$$\widehat{\mathbb{S}} := \{\widehat{S} \in C([t_0, t_{q+1}]) \mid D\widehat{S} \in \mathbb{S}\},$$

where D stands for the (possibly left or right) ordinary differentiation.

Moreover, as soon as (ii) *holds, we also have:*

(iii) *in the spline space $\widehat{\mathbb{S}}$ as well as in any restriction of $\mathbb{S}$ to a subinterval, as well as in any spline space (with $(n+1)$-dimensional section spaces) containing $\widehat{\mathbb{S}}$, any Hermite interpolation problem has a unique solution if and only if the corresponding Schoenberg-Whitney conditions are satisfied.*

Remark 2.2. The main purpose of this paper is interpolation, not blossoms. It is why we will limit ourselves to explaining the precise meaning of (ii) in the special case addressed in Section 3. At this stage we shall just stress that blossoms in the space $\widehat{\mathbb{S}}$ are functions of n variables, defined on a symmetric subset of $[t_0, t_{q+1}]^n$, whose values are obtained by intersecting convenient osculating flats. The interesting thing is that, in the context used to state Theorem 2.1, as soon as they exist, blossoms satisfy a few crucial properties which make it possible to develop the analogue of all design algorithms known for polynomial splines (evaluation, subdivision, ...), and by the way, they automatically generate important tools such as B-spline-type bases, recurrence relations, ... The latter tools are strongly involved in the proof of Theorem 2.1 (see [5]). For further acquaintance with blossoms the reader can refer to [11], [13] for instance.

Remark 2.3. The spaces spanned by the four functions $1, x, (1-x)^p, x^q$, considered in Figures 1 and 2 are not Extended Chebyshev spaces on the interval $[0, 1]$ where we consider them, except in the case $p = q = 3$ where the corresponding space is the polynomial space of degree 3. Indeed, in all other cases, Taylor interpolation is not always possible because there exist non-zero functions with too many zeros at 0 or 1. As a matter of fact, in the definition of $\mathbb{S}$, we can weaken our assumption (a) by assuming that any Hermite interpolation problem in n data based on at least two distinct points has a unique solution in $\mathbb{E}_k$, thus excluding Taylor interpolation. For such a space we shall use the terminology

Quasi Extended Chebyshev space. Equivalently, this means that any nonzero element of $\mathbb{E}_k$ vanishes at most $(n-1)$ times in $[t_k, t_{k+1}]$, but now counting multiplicities up to $(n-1)$. A simple way to guarantee that, as soon as blossoms exist in the spline space $\widehat{\mathbb{S}}$, they satisfy the same expected properties as in the case of Extended Chebyshev sections, is to require each $\mathbb{E}_k$ to be composed of analytic functions. Theorem 2.1 remains valid in this new context too provided that, firstly, all interpolation points are involved at most either $(n-1)$ times (interpolation in the space $\mathbb{S}$) or n (in $\widehat{\mathbb{S}}$), and secondly, all knots of the spline space have positive multiplicities.

3 Interpolation in spline spaces with 4-dimensional sections

The validity of the Schoenberg-Whitney conditions to ensure interpolation in the space $\widehat{\mathbb{S}}$ was proved in [12] under a requirement of total positivity on the connection matrices in the case where the sections are Extended Chebyshev spaces. For polynomial sections this total positivity assumption concerns the matrices connecting the left and right ordinary derivatives at each knot. For sections in more general Extended Chebyshev spaces it concerns the matrices obtained when expressing the joints at a knot via generalised derivatives, defined by means of weight functions (see [1]). The main inconvenience of such an assumption of total positivity is that it is not intrinsic. Indeed, it depends on the chosen differential operators associated with the Extended Chebyshev spaces (see [15]), and there may be many different such operators associated with one given Extended Chebyshev space.

Our condition (ii) of Theorem 2.1 is significantly weaker than total positivity of all connection matrices. It also has the advantage of being intrinsic. However in practice it is not obvious to know when exactly blossoms do exist in a spline space as general as we described in Section 2. This is due to the difficulty of checking that various osculating flats intersect at a single point in high dimension. In [8] we focussed on the somewhat easier special case of four-dimensional sections ("cubic" splines). There, we were able to achieve a necessary and sufficient condition for existence of blossoms. Before stating it, we need firstly to describe the exact framework, and secondly to introduce some geometric characteristics of the spaces used to build our splines.

As in the previous section we consider a subdivision $t_0 < t_1 < \cdots < t_q < t_{q+1}$. From now on we assume the knots to be simple, that is, we are now working with the knot vector

$$\mathcal{K} = (t_1, \ldots, t_q).$$

For $0 \leq k \leq q$, we suppose that $\mathbb{E}_k$ is a three-dimensional Extended Chebyshev space on $[t_k, t_{k+1}]$ or an analytic Quasi Extended Chebyshev space on the latter interval (see Remark 2.3). Let us denote by $\widehat{\mathbb{E}}_k \subset C([t_k, t_{k+1}])$ the four-dimensional space obtained by integration of $\mathbb{E}_k$. On the other hand, for $1 \leq k \leq q$, we choose three real numbers $\beta_1^k, \beta_2^k, \beta_3^k$, with $\beta_1^k, \beta_3^k > 0$. The corresponding Chebyshevian spline space $\widehat{\mathbb{S}}$ is composed of all continuous functions $\widehat{S} : [t_0, t_{q+1}] \to \mathbb{R}$ such that

(1) for $0 \leq k \leq q$ there exists $\widehat{F}_k \in \widehat{\mathbb{E}}_k$ such that $\widehat{S}(t) = \widehat{F}_k(t)$ for all $t \in [t_k, t_{k+1}]$;

(2) $\widehat{S}$ satisfies the the connection conditions

$$\big(\widehat{S}'(t_k{}^+), \widehat{S}''(t_k{}^+)\big)^T = \begin{pmatrix} \beta_1^k & 0 \\ \beta_2^k & \beta_3^k \end{pmatrix} .\big(\widehat{S}'(t_k{}^-), \widehat{S}''(t_k{}^-)\big)^T, \quad 1 \leq k \leq q. \tag{10}$$

Let us now fix an integer $k \in \{0, \ldots, q\}$. The space $\widehat{\mathbb{E}}_k$ is an Extended Chebyshev space or a Quasi Extended Chebyshev space on $[t_k, t_{k+1}]$ and it contains constants. Choosing a basis $(\mathbb{1}, \Phi_1^k, \Phi_2^k, \Phi_3^k)$ in $\widehat{\mathbb{E}}_k$, we consider the function $\Phi_k := (\Phi_1^k, \Phi_2^k, \Phi_3^k)$. We know that blossoms exist in the space $\widehat{\mathbb{E}}_k$, the blossom φ_k of Φ_k being defined on $[t_k, t_{k+1}]^3$ by (see [8], [9] and other references therein):

$$\{\varphi_k(x,y,z)\} = \begin{cases} \mathrm{Osc}_2\Phi_k(x) \cap \mathrm{Osc}_2\Phi_k(y) \cap \mathrm{Osc}_2\Phi_k(z) & \text{if } x, y, z \text{ are pairwise distinct,} \\ \mathrm{Osc}_2\Phi_k(x) \cap \mathrm{Osc}_1\Phi_k(y) & \text{if, up to permutation, } z = y \neq x, \\ \mathrm{Osc}_0\Phi_k(x) := \{\Phi_k(x)\} & \text{if } z = y = x. \end{cases} \tag{11}$$

In (11), $\mathrm{Osc}_1\Phi_k(x) := \{\Phi_k(x) + \lambda\Phi_k'(x), \lambda \in \mathbb{R}\}$ is the tangent line at x and $\mathrm{Osc}_2\Phi_k(x) := \{\Phi_k(x) + \lambda_1\Phi_k'(x) + \lambda_2\Phi_k''(x), \lambda_1, \lambda_2 \in \mathbb{R}\}$ is the osculating plane at x.

In particular, this enables us to consider the Bézier points of Φ_k w.r. to (t_k, t_{k+1}), namely, the points

$$P_{k,0} := \Phi_k(t_k),\ P_{k,1} := \varphi_k(t_k, t_k, t_{k+1}),\ P_{k,2} := \varphi_k(t_k, t_{k+1}, t_{k+1}),\ P_{k,3} := \Phi_k(t_{k+1}). \tag{12}$$

From (11) and (12) one can deduce the existence of real numbers $\lambda_{k,i}^+$, $i = 1, 2, 3$, such that

$$\begin{pmatrix} P_{k,1} - P_{k,0} \\ P_{k,2} - P_{k,0} \end{pmatrix} = \begin{pmatrix} \lambda_{k,1}^+ & 0 \\ \lambda_{k,2}^+ & \lambda_{k,3}^+ \end{pmatrix} \begin{pmatrix} \Phi_k'(t_k) \\ \Phi_k''(t_k) \end{pmatrix}, \tag{13}$$

and a similar relation at the other endpoint:

$$\begin{pmatrix} P_{k,2} - P_{k,3} \\ P_{k,1} - P_{k,3} \end{pmatrix} = \begin{pmatrix} \lambda_{k+1,1}^- & 0 \\ \lambda_{k+1,2}^- & \lambda_{k+1,3}^- \end{pmatrix} \begin{pmatrix} \Phi_k'(t_{k+1}) \\ \Phi_k''(t_{k+1}) \end{pmatrix}. \tag{14}$$

The λ's involved in (13) and (14) are geometric characteristic of the space $\widehat{\mathbb{E}}_k$: they do not depend on the basis chosen to define blossoms. They satisfy

$$\lambda_{k,1}^+ > 0,\ \lambda_{k+1,1}^- < 0,\ \lambda_{k,3}^+ > 0,\ \lambda_{k+1,3}^- > 0.$$

Selecting a basis $(\mathbb{1}, \widehat{\Sigma}_1, \ldots, \widehat{\Sigma}_{n+m})$ in the space $\widehat{\mathbb{S}}$, we set $\widehat{\Sigma} := (\widehat{\Sigma}_1, \ldots, \widehat{\Sigma}_{n+m})$. First observe that $\mathrm{Osc}_1\widehat{\Sigma}(x)$ and $\mathrm{Osc}_2\widehat{\Sigma}(x)$ are well-defined at any $x \in [t_0, t_{q+1}]$, even when x is a knot. Indeed, due to the continuity of the spline $\widehat{\Sigma}$ and to the connection conditions (10), at a given knot t_k, $1 \leq k \leq q$, the left and right tangent lines $\mathrm{Osc}_1\widehat{\Sigma}(t_k^-)$ and $\mathrm{Osc}_1\widehat{\Sigma}(t_k^+)$ are equal, and so are the left and right osculating planes $\mathrm{Osc}_2\Sigma(t_k^-)$ and $\mathrm{Osc}_2\Sigma(t_k^+)$. We simply denote them by $\mathrm{Osc}_1\Sigma(t_k)$ and $\mathrm{Osc}_2\Sigma(t_k)$, respectively. On the other hand, we say that a triplet $(x_1, x_2, x_3) \in [t_0, t_{q+1}]^3$ is *admissible* if, for any integer k, $1 \leq k \leq q$, satisfying $\mathrm{Min}(x_1, x_2, x_3) < t_k < \mathrm{Max}(x_1, x_2, x_3)$, there exists at least one $i \in \{1, 2, 3\}$ such that $x_i = t_k$. With this definition, the necessary and sufficient condition for existence of blossoms obtained in [8] can be stated as follows.

Theorem 3.1. *The Chebyshevian spline space being defined as above, the following two properties are equivalent*

(i) *blossoms exist in the space* $\widehat{\mathbb{S}}$*, in the sense that the blossom* σ *of* Σ *can be defined by formulae similar to* (11)*, but this, only on the subset of* $[t_0, t_{q+1}]^3$ *composed of all admissible triplets* (x, y, z)*;*

(ii) *the connection matrices satisfy:*

$$\beta_1^k \; d_k^+ + \beta_2^k - \beta_3^k \; d_k^- \; > \; 0 \; , \quad 1 \leq k \leq q, \tag{15}$$

where the real numbers d_k^- *and* d_k^+ *are defined by*

$$d_k^- := \frac{\lambda_{k,2}^- - \lambda_{k,1}^-}{\lambda_{k,3}^-}, \quad d_k^+ := \frac{\lambda_{k,2}^+ - \lambda_{k,1}^+}{\lambda_{k,3}^+}, \quad 1 \leq k \leq q. \tag{16}$$

To illustrate Theorem 3.1, suppose that, for $0 \leq k \leq q$, the space $\widehat{\mathbb{E}}_k$ is the space spanned by the functions $1, x, (1-x)^{p_k}, x^{q_k}$ on $[0,1]$ (see Figure 1), up to the change of variable $\vartheta(t) := (t - t_k)/(t_{k+1} - t_k)$ (*variable degree polynomial splines with connection matrices*). The corresponding numbers d_k^- and d_k^+ are then given by (see [9]):

$$d_k^- = \frac{1 - q_{k-1}}{t_k - t_{k-1}}, \quad d_k^+ = \frac{p_k - 1}{t_{k+1} - t_k}, \quad 1 \leq k \leq q. \tag{17}$$

Other examples can be found in [8].

In the particular situation considered in this section, we thus know that conditions (15) are equivalent to (i) of Theorem 2.1 and that they imply (iii) of the same theorem. Since "cubic" splines are the most commonly used splines, we will actually focus on the spline space $\widehat{\mathbb{S}}$ rather than on $\mathbb{S}$.

Corollary 3.2. *Suppose that the connection matrices satisfy conditions* (15)*. Then, a given Hermite interpolation problem* (1) *in* $N = q + 4$ *data has a unique solution in the spline space* $\widehat{\mathbb{S}}$ *if and only if the interpolation points* $y_{-3} \leq \cdots \leq y_q \in [t_0, t_{q+1}]$ *satisfy the Schoenberg-Whitney conditions, that is, with the notations introduced in Theorem 2.1, if and only if:*

$$y_{k-4} < t_k < y_k \quad \textit{for } 1 \leq j \leq q.$$

Let us illustrate the latter result in the special case of Lagrange interpolation. Subsequently, we shall use the terminology *Chebyshevian space* for either an Extended Chebyshev space or an analytic Quasi Extended Chebyshev space. Although the result below can be valid without it, as mentioned in Remark 2.3, analyticity is a practical assumption to guarantee blossoms to satisfy all desirable properties in Quasi Extended Chebyshev space.

Corollary 3.3. *Given any real numbers* $x_1 < \cdots < x_N$*, set* $q := N - 4$ *and define the points* $t_0, \ldots, t_{q+1}$ *as follows*

$$t_0 := x_1, \quad t_k := x_{k+2} \quad \textit{for } 1 \leq k \leq q, \quad t_{q+1} := x_N.$$

For $0 \leq k \leq q$, choose a four-dimensional Chebyshevian space $\widehat{\mathbb{E}}_k$ containing constants so that $\mathbb{E}_k := D\widehat{\mathbb{E}}_k$ is a Chebyshevian space on $[t_k, t_{k+1}]$. For $1 \leq k \leq q$, choose real numbers $\beta_1^k, \beta_2^k, \beta_3^k$ so that

$$\beta_1^k, \beta_3^k > 0, \quad \beta_2^k > \beta_3^k \; d_k^- - \beta_1^k d_k^+. \tag{18}$$

where the coefficients d_k^- and d_k^+ are associated with $\widehat{\mathbb{E}}_k$ according to (16)*. Let $\widehat{\mathbb{S}}$ denote the corresponding spline space defined as above. Then, for any $\alpha_1, \ldots, \alpha_N \in \mathbb{R}$, there exists a unique $\widehat{S} \in \widehat{\mathbb{S}}$ such that*

$$\widehat{S}(x_k) = \alpha_k, \quad 1 \leq k \leq N. \tag{19}$$

We thus have constructed infinitely many spline spaces to obtain a unique solution to the Lagrange interpolation problem (19). We now have at our disposal infinitely many shape parameters, namely:

- on the one hand, the shape parameter(s) inherent in the Chebyshevian space $\widehat{\mathbb{E}}_k$, $0 \leq k \leq q$;
- on the other, all coefficients $\beta_1^k, \beta_2^k, \beta_3^k$, $1 \leq k, \leq q$, supposed to meet the requirement (18).

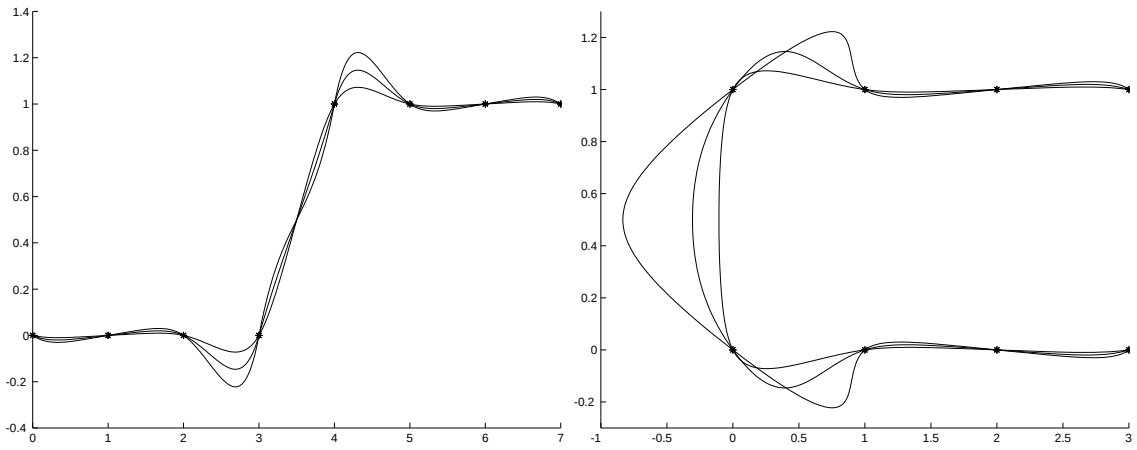


Figure 3: Lagrange interpolation at regularly spaced points. Not-a-knot procedure (that is, interior knots of the spline space at 2,3,4,5). Up to appropriate changes of variable, sections in the space spanned by $1, x, (1-x)^5, x^5$. Everywhere $\beta_1^k = \beta_3^k = 1$ and everywhere $\beta_2^k = 0$, except at the two central knots where $\beta_2^k = -7.9, -6, 0$ (from the most to the least oscillatory).

To illustrate this, let us consider again the case of variable degree polynomial splines with connection matrices adressed in formula (17). In this case, according to the latter formula, in order to satisfy the connection requirements (18), for given positive β_1^k, β_3^k, we have to choose β_2^k as follows:

$$\beta_2^k > -\beta_1^k \frac{p_k - 1}{t_{k+1} - t_k} - \beta_3^k \frac{q_{k-1} - 1}{t_k - t_{k-1}}, \quad 1 \leq k \leq q. \tag{20}$$

Interpolation in the corresponding spline spaces is illustrated in Figure 3. As in all our subsequent examples, the spline space is defined according to the situation described in Corollary 3.3.

Remark 3.4. The latter inequality (20) makes it obvious that total positivity is not at all necessary to be able to interpolate under Schoenberg-Whitney conditions (see Figure 3; also, see the example of trigonometric splines in Remark 3.6 below). Indeed, in this case, total positivity of the connection matrix at the knot t_k just means that the non-diagonal coefficient β_2^k is non-negative. As an instance, for cubic polynomial splines with connection matrices, and for regularly spaced knots with $t_{k+1} - t_k = 1$, our condition (20) becomes

$$\beta_2^k > -2(\beta_1^k + \beta_3^k), \quad 1 \leq k \leq q.$$

Remark 3.5. One of the possible interests of the shape parameters introduced is to avoid oscillations, that is, to try and ensure shape preservation (monotonicity of the interpolating function if the data are monotonically distributed or monotonicity in one direction of the interpolating curve if the data points are monotonically distributed in that direction, ...). This can be obtained with a relevant choice for the Chebyshevian spaces used to define $\widehat{\mathbb{S}}$ and for their inherent shape parameter(s). This is illustrated in Figures 4 and 5 with C^2 variable degree polynomial splines in the case $p = q$. We can see the efficiency of the tension shape parameter p as it goes to $+\infty$. Note that this is the very reason why C^2 variable degree polynomial splines were introduced (see [2]).

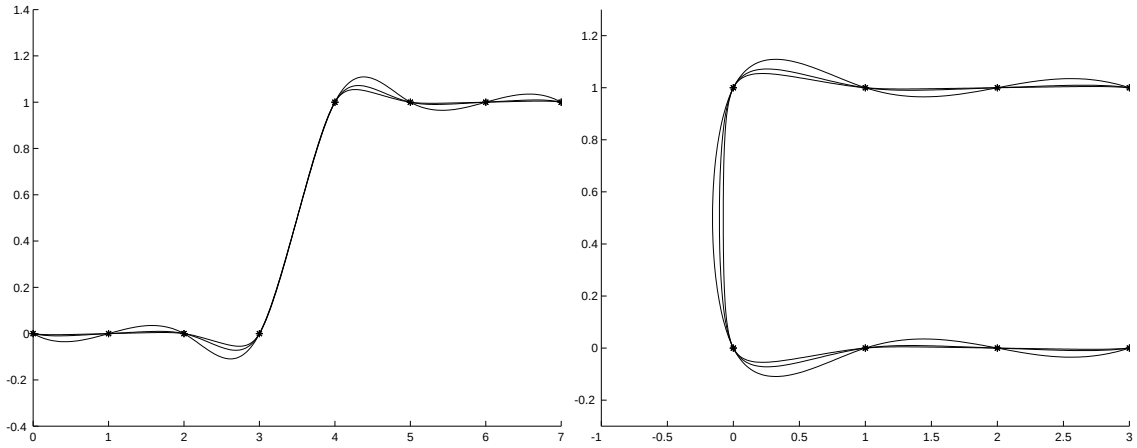


Figure 4: Tension effects with Lagrange interpolation by C^2 variable degree polynomial splines: regularly spaced interpolation points, not-a-knot procedure, and (up to changes of variable) sections in the space spanned by $1, x, (1-x)^p, x^p$, with, from the most to the least oscillatory $p = 3$ ($\mathbb{P}_3$);5;7.

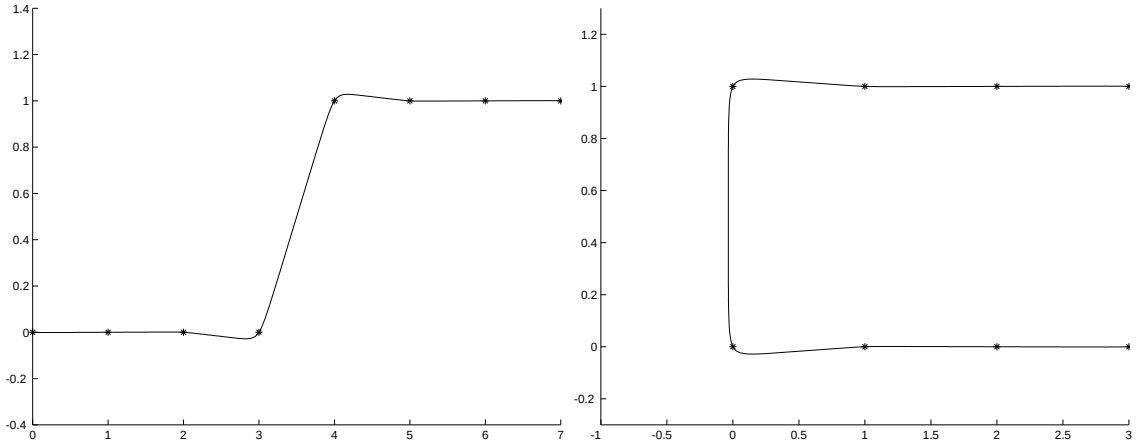


Figure 5: The same as in Figure 4, with $p = 15$.

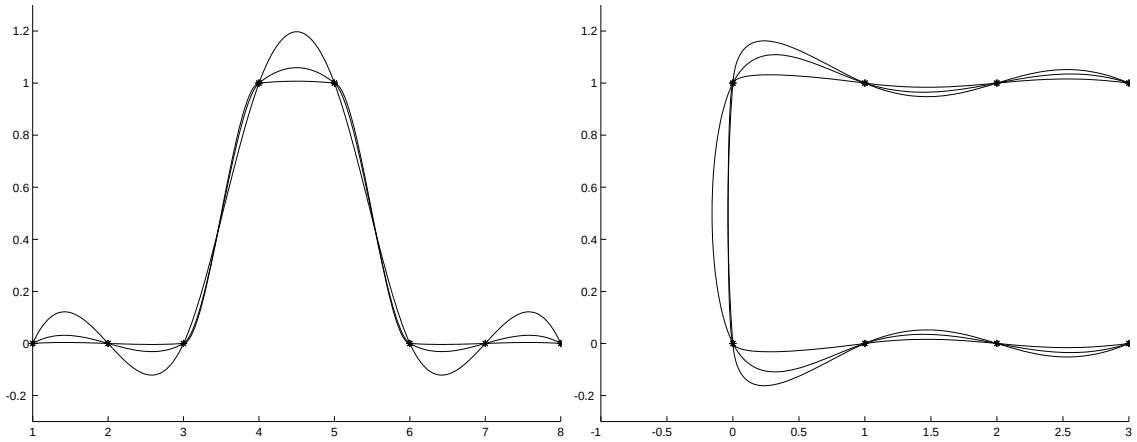


Figure 6: Lagrange interpolation by polynomial splines with connection matrices. Regularly spaced interpolation points and not-a-knot procedure. Everywhere $\beta_1^k = 1$ and $\beta_2^k = 0$. Left: from the most to the least oscillatory $\beta_3^1 = \beta_3^3 = 1; 10; 100$ and $\beta_3^2 = \beta_3^4 = 1; 0.1; 0.01$. Right: from the most to the least oscillatory, in the first variable $(\beta_3^1, \beta_3^2, \beta_3^3, \beta_3^4) = (1,1,1,1); (1,0.1,10,1); (1,0.1,10,1)$ and in the second one $(\beta_3^1, \beta_3^2, \beta_3^3, \beta_3^4) = (1,1,1,1); (1,0.1,10,1); (0.01,10,0.1,100)$.

Similar effects can be obtained with the help of connection matrices. This is illustrated in Figure 6 for cubic polynomial splines with connection matrices satisfying $\beta_1^k = 1$, $\beta_2^k = 0$ everywhere. In that case too, efficients tension effects can be obtained by choosing suitable positive values for the coefficients β_3^k .

Remark 3.6. Depending on the given interpolation points $x_1, \ldots, x_N$ and on our choice of the Chebyshevian spaces $\widehat{\mathbb{E}}_k$ it may be impossible to interpolate under C^2 continuity. This will make absolutely necessary to introduce connection matrices. For instance, when choosing $\widehat{\mathbb{E}}_k$ as the *trigonometric space* spanned by $1, x, \cos x, \sin x$, over $[t_k, t_{k+1}]$, for blossoms to exist in the corresponding C^2 spline space it is necessary and sufficient to require $\max(t_{k+2} - t_k) < 2\pi$. Otherwise, we have to use connexion matrices satisfying condition (4.14) of [8].

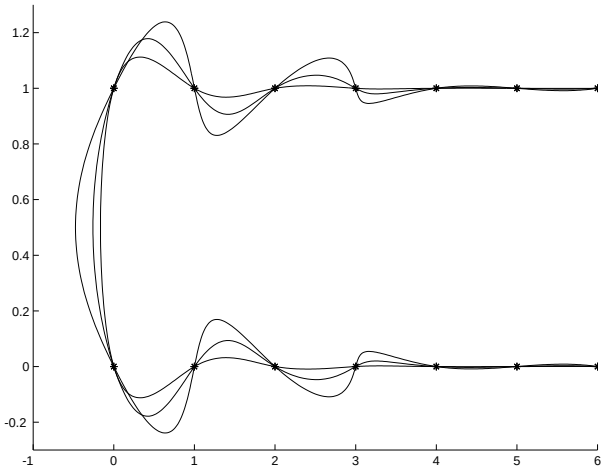


Figure 7: C^2 Lagrange interpolating trigonometric spline curves with, for $0 \leq k \leq 10$, from the least to the most oscillatory, $t_{k+1} - t_k = 1$; 2.8; 3.1. Sequence of interpolation points: $(y_{-3}, \ldots, y_{10}) = (t_0, \frac{1}{2}(t_0 + t_1), t_1, \ldots, t_{10}, \frac{1}{2}(t_{10} + t_{11}), t_{11})$

In case we want to construct a parametric interpolating trigonometric spline curve (for given points $\alpha_k \in \mathbb{R}^d$), it may be interesting to take the length of the intervals as shape parameters. This is illustrated in Figure 7 with C^2-continuity and regularly spaced knots t_k, $0 \leq k \leq q + 1 := 11$. To be in keeping with the situation described in Corollary 3.3, we interpolate at each knot (including the end points) and at two additional interpolation points in the interior of the first and last interval, namely $\frac{1}{2}(t_0 + t_1)$ and $\frac{1}{2}(t_{10} + t_{11})$. Such a choice for the two additional interpolation points implies reduced oscillations towards the end points. According to (iii) of Theorem 2.1, $\max(t_{k+2} - t_k) < 2\pi$ is only a sufficient condition for ensuring existence and unicity of any Lagrange interpolation problem in a C^2-trigonometric spline space. To compare, the previous existing sufficient condition was existence of B-splines with supports contained in an interval on which the trigonometric space is an Extended Chebyshev space, that is, $\max(t_{k+4} - t_k) < 2\pi$. For instance, in

Figure 7, only the first length fits in with the latter condition, which itself can be much better than the total positivity requirement. By way of example, in [8] we pointed out that C^2 trigonometric splines are never in keeping with total positivity when using some natural weight functions associated with the trigonometric space.

Remark 3.7. Shape effects and shape parameters play an important rôle in Geometric Design. For interpolating curves, they are mainly used for shape preservation purposes as explained above. But why not also use them to play with the curves for "design interpolation" purposes. This would require to learn how to modify the various shape parameters depending on the shape effects we wish to obtain.

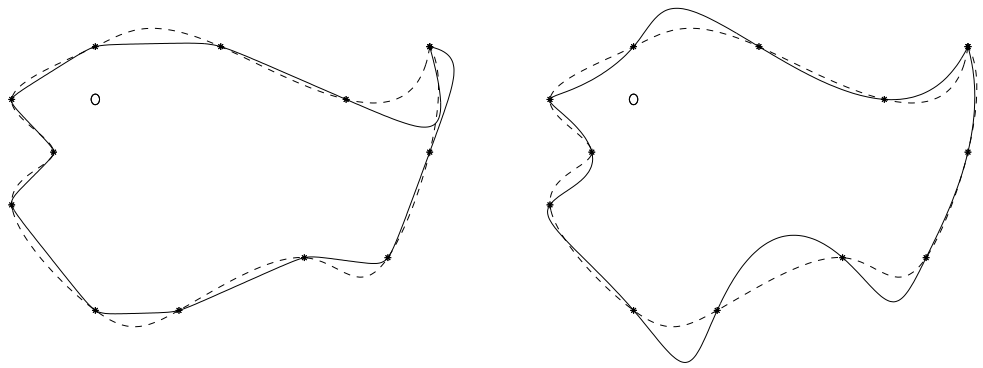


Figure 8: Shape effects with Lagrange interpolation by C^2 splines. Regularly spaced knots. Left: hyperbolic fish with $t_{k+1} - t_k = 0.1$ (dotted line: nearly polynomial splines); 14 (continuous line). Right: trigonometric fish with $t_{k+1} - t_k = 0.1$ (dotted line: nearly polynomial splines); 3.

To illustrate this we shall limit ourselves to C^2 splines with sections either in the trigonometric space (see Remark 3.6) or in the hyperbolic space spanned by the four functions $1, x, \cosh x, \sinh x$ which is an Extended Chebyshev space over the whole real line. Roughly speaking, when varying the length between knots, interpolation with either trigonometric spline curves (with the limitation $t_{k+2} - t_k < 2\pi$) or hyperbolic spline curves (with no limitation) provides us with somewhat opposite shape effects, as shown in Figure 8 for regularly spaced knots. In both cases, as well as in Figure 9 later, the interpolation points are all the knots (including the end points) and the middles of the first and last interval. It seems obviously interesting to combine these shape effects by mixing hyperbolic and trigonometric sections. A tiny idea of the possibilities is given in Figure 9, where, for the sake of uniformity we limit ourselves to regularly spaced knots in all examples. Many more effects can be obtained by cancelling the latter restriction, of course within existence of blossoms, that is, according to (18) (see relations (4.14) and (4.18) of [8]). Allowing connection matrices or using other kinds of Chebyshevian spaces for the sections (e.g., those involved in Figure 2, right) would also increase the possibilities.

Remark 3.8. To conclude, note that we have assumed all interior knots to be simple, but it is actually possible to allow double knots. At a double knot t_k, in the definition of the

spline space $\widehat{\mathbb{S}}$ the connection condition reduces to

$$\widehat{S}'(t_k^+) = \beta_1^k \ \widehat{S}'(t_k^-)$$

for some positive β_1^k. No additional condition is required either by interpolation in $\mathbb{S}$ or by existence of blossoms in $\widehat{\mathbb{S}}$.

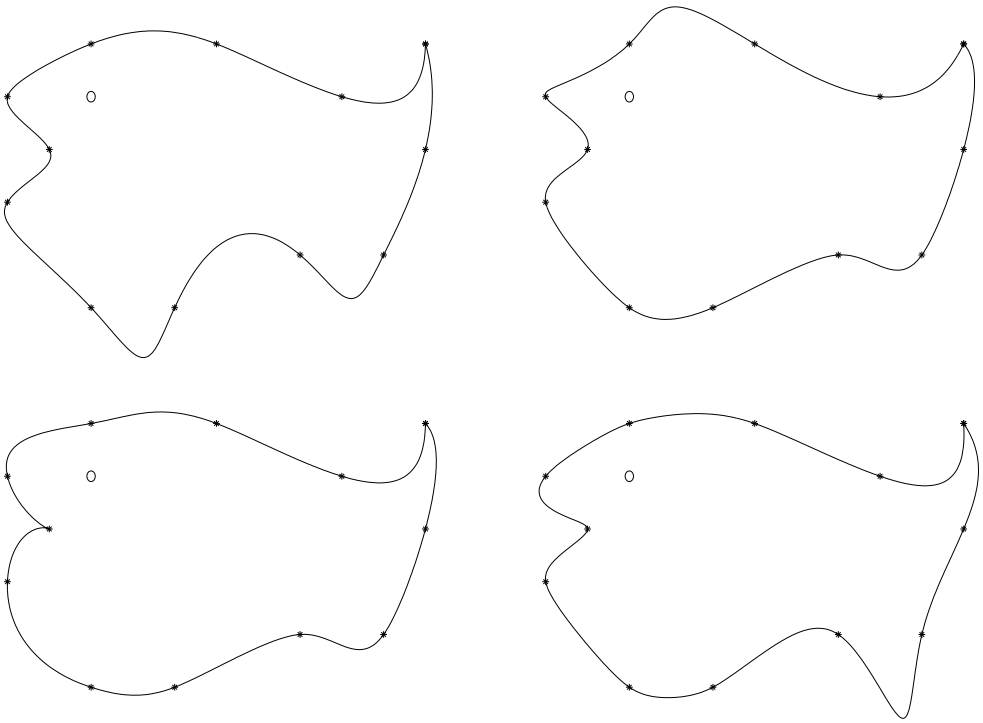


Figure 9: Shape effects with Lagrange interpolation by C^2 mixed hyperbolic/trigonometric spline curves, starting from and ending at the top of the tail, first back, then belly. Regularly spaced knots. Upper left: 4H 6T, $t_{k+1} - t_k = 3$. Upper right: 4T 6H, $t_{k+1} - t_k = 3$. Lower left: 2H 4T 4H, $t_{k+1} - t_k = 3$. Lower right: 3H 1T 4H 1T 1H, $t_{k+1} - t_k = 4.5$ (H=hyperbolic section, T=trigonometric section).

References

[1] Barry, P.J., de Boor-Fix dual functionals and algorithms for Tchebycheffian B-splines curves, *Constr. Approx.* **12** (1996), 385–408.

[2] P. Costantini, On monotone and convex spline interpolation, *Math. Comput.* **46** (1986), 203–214.

[3] N. Dyn and C.A. Micchelli, Piecewise polynomial spaces and geometric continuity of curves, Numer. Math. **54** (1988), 319-337.

[4] S. Karlin and Z. Ziegler, Tchebysheffian spline functions, *SIAM J. Numer. Anal.*, Series B, **3** (1966), 514–543.

[5] A. Kayumov and M.-L. Mazure, Chebyshevian splines: interpolation and blossoms, preprint.

[6] P. E. Koch and T. Lyche, Exponential B-splines in tension, in *Approximation Theory VI*, C. K. Chui, L. L. Schumaker, and J. D. Ward (eds), Acad. Press, N. Y., 1989, 361–364.

[7] T. Lyche and L .L. Schumaker, Total positivity properties of LB-splines, in *Total Positivity and Its Applications*, M. Gasca and C. Micchelli (eds.), Kluwer , Dodrecht, 1996, pp. 35-46

[8] M.-L. Mazure, Chebyshev splines beyond total positivity, *Adv. Comput. Math.* **14** (2001), 129-156.

[9] M.-L. Mazure, Quasi-Chebyshev splines with connection matrices. Application to variable degree polynomial splines, *Computer-Aided Geom. Design* **18** (2001), 287-298.

[10] M.-L. Mazure, On the equivalence between existence of B-spline bases and existence of blossoms, *Constr. Approx.* **20** (2004), 603–624.

[11] M.-L. Mazure, Ready-to-blossom bases in Chebyshev spaces, in *Topics in Multivariate Approximation and Interpolation*, K.Jetter, M.Buhmann, W.Haussmann, R.Schaback, and J.Stoeckler (eds), Elsevier, 12, chapitre 6, 2005.

[12] G. Mühlbach, One sided Hermite interpolation by piecewise different generalized polynomials, to appear in *Adv. Comput. Math.*.

[13] H. Pottmann, The geometry of Tchebycheffian splines, *Computer-Aided Geom. Design* **10** (1993), 181–210.

[14] I. J. Schoenberg and A. Whitney, On Pólya frequency functions, III, *Trans. Amer. Math. Soc.* **74** (1953), 246–259.

[15] L .L. Schumaker, *Spline Functions*, Wiley Interscience, N.Y., 1981.

Marie-Laurence Mazure
LMC-IMAG
Université Joseph Fourier
mazure@imag.fr
http://www-lmc.imag.fr/lmc-cf/Marie-Laurence.Mazure/

Computing Roadmaps in Smooth Real Algebraic Sets

Marc Mezzarobba Mohab Safey El Din

Abstract

Let $(f_1, \ldots, f_s)$ be polynomials in $\mathbb{Q}[X_1, \ldots, X_n]$ of degree bounded by D that generate a radical equidimensional ideal of dimension d and let $\mathcal{V} \subset \mathbb{C}^n$ be the locus of their complex zero set which is supposed to be smooth. A *roadmap* in $\mathcal{V} \cap \mathbb{R}^n$ is a real algebraic curve contained in $\mathcal{V} \cap \mathbb{R}^n$ which has a non-empty and connected intersection with each connected component of $\mathcal{V} \cap \mathbb{R}^n$.

The classical strategy to compute roadmaps is due to J. Canny and leads to algorithms having a complexity within $D^{\mathcal{O}(n^2)}$ arithmetic operations in $\mathbb{Q}$. This strategy is based on computing a polar variety of dimension 1 and a recursion on the studied variety intersected with fibers taken above a critical value of a projection. Thus, it requires computations with real algebraic numbers and introduces singularities at each recursive call. Thus, no efficient implementation of roadmap algorithms have been obtained until now. Our aim is to provide an efficient implementation of the roadmap algorithm. We show how to slightly modify this strategy in order to avoid the use of real algebraic numbers and to deal with smooth algebraic sets at each recursive call in the case where the input variety is smooth. Our complexity is $h^d D^{\mathcal{O}(n)}$ operations in $\mathbb{Q}$ where h bounds the number of recursive call in our algorithm. This quantity is related to the geometry of $\mathcal{V} \cap \mathbb{R}^n$ and is bounded by $D^{\mathcal{O}(n)}$, thus in worst cases our algorithm has a complexity within $D^{\mathcal{O}(n^2)}$ arithmetic operations. We report on some experiments done with a preliminary implementation of our algorithm.

Keywords. Polynomial System Solving, Real Solutions, Connectedness, Complexity.

Introduction

Let $(f_1, \ldots, f_s)$ be polynomials in $\mathbb{Q}[X_1, \ldots, X_n]$ of degree bounded by D that generate a radical equidimensional ideal of dimension d. Let $\mathcal{V} \subset \mathbb{C}^n$ be the algebraic set defined by $f_1 = \cdots = f_s = 0$ which is supposed to be smooth in the sequel. A *roadmap* $\mathcal{R}$ associated to $\mathcal{V} \cap \mathbb{R}^n$ is an algebraic curve contained in $\mathcal{V}$ having a *non-empty* and *connected* intersection with each connected component of $\mathcal{V} \cap \mathbb{R}^n$. This paper is devoted to design an efficient algorithm computing roadmaps in smooth real algebraic sets leading to an efficient implementation.

Computing roadmaps allows to reduce general connectivity decision problems to connectivity decision in dimension 1 for which there exist algorithms (see [4]). The problem of deciding connectivity is motivated by problems arising in robot motion planning where deciding if two given points belong to the same connected component of a semi-algebraic set

is a question of first importance (see [20]). In this context, the computation of a *roadmap* can be seen as a preliminary step (see [3]) using connecting subroutines between these points and the computed roadmap. Roadmaps can also be used to compute the number of connected components of a semi-algebraic set and parametrized versions of some roadmap algorithms are used to obtain a semi-algebraic description of the connected components of a semi-algebraic set (see [4]).

Roadmaps can be extracted from a cylindrical algebraic decomposition but such an approach leads to algorithmic solutions which are doubly exponential in the number of variables. The notion of roadmap is explicitly introduced in 1987 by J. Canny (see [5, 6]) who provides an algorithm computing roadmaps for semi-algebraic set in $k^n D^{\mathcal{O}(n^4)}$ operations in $\mathbb{Q}$ (where k is the number of inequalities defining the considered semi-algebraic set. In the case of real algebraic sets we are considering here, the complexity of his algorithm is $D^{\mathcal{O}(n^4)}$. A Monte-Carlo version of Canny's algorithm computes roadmaps in $D^{\mathcal{O}(n^2)}$. Further developments can be found in [14, 13, 15] and in particular [3] where an algorithm computing roadmaps in a semi-algebraic set having a complexity $k^{d'} D^{\mathcal{O}(n^2)}$ where k is the number of inequalities and d' the dimension of the considered semi-algebraic set. In the case of real algebraic sets, the latter algorithm has a complexity $D^{\mathcal{O}(n^2)}$ arithmetic operations in $\mathbb{Q}$. All the algorithms cited above are, in the case of real algebraic sets, based on Canny's strategy we present below.

Canny's strategy to compute roadmaps. Suppose that $\mathcal{V} \cap \mathbb{R}^n$ is compact. Canny's algorithm computes a "silhouette", i.e. the critical locus of the restriction of a projection on a plane P to $\mathcal{V}$. Since $\mathcal{V} \cap \mathbb{R}^n$ is compact, this silhouette has a non-empty intersection with each connected component of $\mathcal{V} \cap \mathbb{R}^n$ but this intersection may be non connected. Classical results of Morse theory show that it is sufficient to construct roadmaps in the slices which are fibers taken above the critical values of the restriction to the considered silhouette of the projection on a line lying in P to obtain connected algebraic curves in each connected component of $\mathcal{V} \cap \mathbb{R}^n$. Figure 1 illustrates this process in the case of a torus in $\mathbb{R}^3$.

The construction of these roadmaps in each slice is done by considering once again a silhouette in each slice and the critical values of some projection on a line restricted to the new silhouette. Once again one has to construct roadmaps in each fiber taken above these critical values and so on until the considered slices are 1-dimensional.

Thus the above construction is based on the recursive calls to a procedure computing:

- a critical locus of the projection on the plane (X_1, X_2);
- the set E of critical values of a projection on X_1

on the set of polynomials defining $\mathcal{V}$ where X_1 is instantiated to v for each v in E and the set of variables $(X_1, \ldots, X_n)$ is obviously replaced by $(X_2, \ldots, X_n)$ so that the next silhouette is computed relatively to the plane (X_2, X_3).
Remark that each slice defines a singular variety and is defined as a pre-image of a real algebraic number by some projection. Thus, deformation techniques based on the introduction of infinitesimals are required to deal with these singular varieties. This, and the use

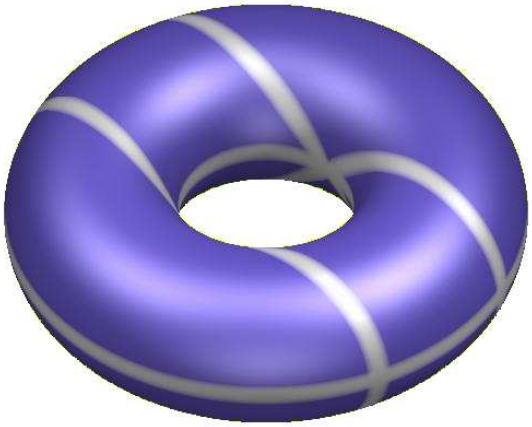


Figure 1: Canny's roadmap in the case of a torus.

of real algebraic numbers, makes the arithmetic on which the computations are performed extremely heavy. Thus, in despite to the rather good complexity of roadmap algorithms, they had never been implemented. In this paper, we show how to slightly modify the above geometric procedure to avoid the aforementioned problems in the case where the input polynomials define a smooth algebraic variety.

Slight modification of Canny's strategy in smooth situations. As above, we suppose $\mathcal{V} \cap \mathbb{R}^n$ is compact and we make the following assumption:

- (H_1) the critical locus C of the projection on (X_1, X_2) restricted to $\mathcal{V}$ is 1-dimensional and is smooth.
- (H_2) Above each critical value of the projection on X_1 restricted to C, there exists a unique critical point.

In this case, one can connect the connected components of C lying in the same connected components of $\mathcal{V} \cap \mathbb{R}^n$ by considering fibers *between* each critical value of the projection on X_1 restricted to C. Figure 2 illustrates this process.

Thus, our algorithm is based on the recursive call of the following procedure:

- compute the critical locus C of the projection on the plane (X_1, X_2);
- compute a set $E \subset \mathbb{Q}$ containing a rational number between each critical value of the projection on X_1 restricted to C.

on the set of input polynomials defining $\mathcal{V}$ where X_1 is replaced by v for each $v \in E$.

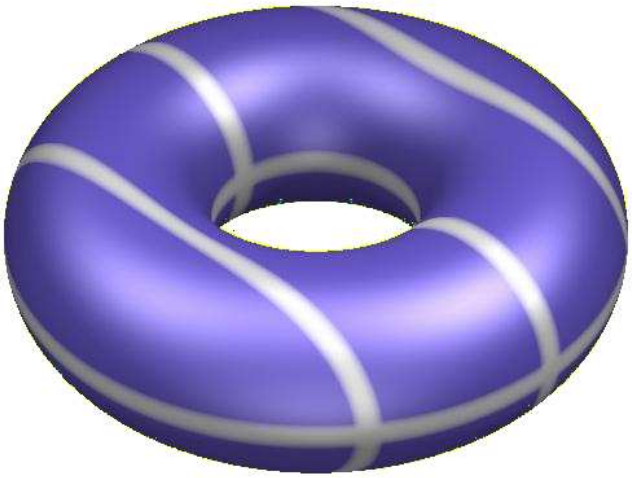


Figure 2: Roadmap obtained by modifying Canny's strategy in the case of a torus.

Remark now that since the fibers are taken above a regular value of the projection on X_1, the slices are smooth algebraic varieties. Moreover, the computations are no longer performed using real algebraic numbers since one can take the fibers above rational numbers.

The proof of the connectedness of the roadmap constructed with respect to the above scheme closely follows the one of Canny's algorithm.

Plan of the paper. In the next section, we show how to reduce the study of smooth real algebraic sets to the study of compact smooth real algebraic sets and that the assumption which has been done in the modification of Canny's strategy is *generically* satisfied. The following section is devoted to the formal description of the algorithm we obtain. In the last section, we study the complexity of our contribution using Lecerf's results (see [12, 16]) on Geometric Resolution and present some results obtained with our implementation which is based on Gröbner bases computations.

1 Preliminaries

This section is devoted to some preliminaries required in the sequel. We consider here a set of polynomials $(f_1, \ldots, f_s)$ in $\mathbb{Q}[X_1, \ldots, X_n]$ generating a radical equidimensional ideal of dimension d and their complex zero set $\mathcal{V} \subset \mathbb{C}^n$ which is supposed o be smooth.

We first show how to reduce the problem of computing a roadmap in a smooth real algebraic set to computing a roadmap in a smooth and *compact* real algebraic set.

Theorem 1.1. *Let $A = (a_1, \ldots, a_n) \in \mathbb{Q}^n$ and E_A be the set of critical values of the square of the euclidean distance to A restricted to $\mathcal{V}$. Let R be a rational number which is greater*

that $\max(e, e \in E)$ *and* X_{n+1} *be a new variable. Consider the algebraic variety* $\mathcal{W}_A$ *defined by:*

$$f_1 = \cdots = f_s = ((X_1 - a_1)^2 + \cdots + (X_n - a_n)^2 + X_{n+1}^2 - R^2)$$

There exists a Zariski-closed subset $\mathcal{A} \subset \mathbb{C}^n$ *such that for all* $A \in \mathbb{Q}^n \setminus \mathcal{A}$*, the following holds:*

- *the above polynomial system generates a radical and equidimensional ideal,*
- $\mathcal{W}_A$ *is smooth,*
- $\mathcal{W}_A \cap \mathbb{R}^{n+1}$ *is compact.*

Moreover, the projection of a roadmap computed in $\mathcal{W} \cap \mathbb{R}^{n+1}$ *onto* $(X_1, \ldots, X_n)$ *is a roadmap on* $\mathcal{V} \cap \mathbb{R}^n$*.*

Proof. We only provide here a sketch of proof. The existence of the Zariski-closed subset $\mathcal{A}$ such that for all $A = (a_1, \ldots, a_n) \in \mathbb{Q}^n \setminus \mathcal{A}$, the system

$$f_1 = \cdots = f_s = ((X_1 - a_1)^2 + \cdots + (X_n - a_n)^2 + X_{n+1}^2 - R^2)$$

generates a radical equidimensional ideal and defines a smooth algebraic variety comes from its characterization as a set of critical values of a polynomial mapping. We refer to [1] and [2] for similar reasonings.

The compacity of $\mathcal{W}_A$ is obvious. The properties of connectedness of the projection of a roadmap in $\mathcal{W}_A$ are done in [4, Chapter 15]. □

We consider now $\mathbf{A} \in GL_n(\mathbb{Q})$ and, given $f \in \mathbb{Q}[X_1, \ldots, X_n]$, we denote by $f^{\mathbf{A}}$ the polynomial $f(\mathbf{A}.X)$ where X denotes the vector $(X_1, \ldots, X_n)$. We denote by $\mathcal{V}^{\mathbf{A}}$ the algebraic variety defined by:

$$f_1^{\mathbf{A}} = \cdots = f_s^{\mathbf{A}} = 0.$$

Consider also the canonical projections Π_i:

$$\begin{array}{rccc} \Pi_i : & \mathbb{C}^n & \longrightarrow & \mathbb{C}^i \\ & (x_1, \ldots, x_n) & \to & (x_1, \ldots, x_i) \end{array}$$

and denote by $K(\Pi_i, \mathcal{V}^{\mathbf{A}})$ the critical locus of Π_i restricted to $\mathcal{V}^{\mathbf{A}}$.

Theorem 1.2. *[2] If* $\mathcal{V}$ *is smooth and equidimensional, there exists a Zariski-closed subset* $\mathcal{A} \subset GL_n(\mathbb{C})$ *such that for all* $\mathbf{A} \in GL_n(\mathbb{Q}) \setminus \mathcal{A}$*, for all* $i \in \{1, \ldots, d-1\}$*,* $K(\Pi_i, \mathcal{V}^{\mathbf{A}})$ *are smooth equidimensional algebraic varieties.*

From the above result, one deduces easily the following one:

Corollary 1.3. *Up to a generic linear change of coordinates* $\mathbf{A} \in GL_n(\mathbb{Q})$*, for* $j = 1, \ldots, d-2$*, there exists a Zariski-closed subset* $\mathcal{P}_j$ *such that for all* $(p_1, \ldots, p_j) \in \mathbb{Q}^j \setminus \mathcal{P}_j$*,* $K(\Pi_{j+2}, \mathcal{V}^{\mathbf{A}}) \cap \Pi_j^{-1}(p_1, \ldots, p_j)$ *is a smooth algebraic curve.*

Remark 1.4. The above corollary implies that at each recursive call, the assumption of smoothness of the 1-dimensional computed critical locus and on the computed fibers are satisfied generically.

Proposition 1.5. *Up to a generic linear change of coordinates* $\mathbf{A} \in GL_n(\mathbb{Q})$*, for* $j = 1, \ldots, d-2$*, there exists a Zariski-closed subset* $\mathcal{P}_j$ *such that for all* $(p_1, \ldots, p_j) \in \mathbb{Q}^j \setminus \mathcal{P}_j$*, there exists exactly one critical point above each critical value of* $\pi_{j+2} : (x_1, \ldots, x_n) \to x_{j+2}$ *restricted to* $K(\Pi_{j+2}, \mathcal{V}^{\mathbf{A}}) \cap \Pi_j^{-1}(p_1, \ldots, p_j)$*.*

Proof. Applying Corollary 1.3, for all $j = 1, \ldots, d-2$, $K(\Pi_{j+2}, \mathcal{V}^{\mathbf{A}}) \cap \Pi_j^{-1}(p_1, \ldots, p_j)$ is smooth.

Consider first the case $j = d-2$ and suppose we are dealing with hypersurfaces. Then, the result is a consequence of [4, Proposition 7.9]. When $j = d-2$ and we are dealing with a general algebraic set $\mathcal{V}^{\mathbf{A}}$ defined by $f_1^{\mathbf{A}} = \cdots = f_s^{\mathbf{A}} = 0$, the result is obtained by remarking that the critical points of a generic projection π restricted to $\mathcal{V}^{\mathbf{A}}$ are contained in the set of limits of the critical points of π restricted to the hypersurface defined by $f_1^{\mathbf{A}\,2} + \cdots + f_s^{\mathbf{A}\,2} - \varepsilon$ when ε tends to 0 (see e.g.[5]).

The other cases are obtained by eventually changing $\mathbf{A}$ in such a way that the variables $X_1, \ldots, X_{j+1}$ are the only one which are changed so that the previous critical loci of higher dimension are not changed and one recover a generic situation for the critical loci of lower dimension. □

2 Roadmap Algorithm

We suppose in the sequel that the assumptions (H_1) and (H_2) are satisfied.

The proof of our algorithm is based on the following proposition which is a well-known result of Morse Theory. Since we have not found a reference where it is explicitely proved, we provide a sketch of proof.

Proposition 2.1. *Let* $[a, b] \subset \mathbb{R}$ *containing a unique critical value* v *of* Π_1 *restricted to* $K(\Pi_2, \mathcal{V})$ *and* $D_{[a,b]}$ *be a connected component of* $\big(\mathcal{V} \cap \Pi_1^{-1}([a, b])\big) \cap \mathbb{R}^n$*. Then, the semi-algebraic set* $D_{[a,b]} \cap \big(\Pi_1^{-1}(a) \cup K(\Pi_2, \mathcal{V}) \cup \Pi_1^{-1}(b)\big)$ *is connected.*

Proof. Let x and y be two distinct points of $D_{[a,b]} \cap \big(\Pi_1^{-1}(a) \cup K(\Pi_2, \mathcal{V}) \cup \Pi_1^{-1}(b)\big)$. Suppose first $\Pi_1(x) < v < \Pi_1(y)$. since Π_1 realizes a locally trivial fibration on $D_{[a,v[}$ (resp. $D_{]v,b]}$) x (resp. y) can be connected to $D_{[a,b]} \cap \Pi_1^{-1}(a)$ (resp. $D_{[a,b]} \cap \Pi_1^{-1}(b)$ via a continuous path. Let x' and y' be the respective intersection of these continuous path with $\Pi_1^{-1}(a)$ and $\Pi_1^{-1}(b)$. Since $\mathcal{V}$ is smooth, there exists a path $\gamma : t \in [0, 1] \to D_{[a,b]}$ linking x' and y' which does not contain the unique critical point p such that $\Pi_1(p) = v$. We then exhibit a path contained in $K(\Pi_2, \mathcal{V}) \cap D_{[a,b]}$ by studying $\Pi_1^{-1}(\Pi_1(\gamma(t))) \cap K(\Pi_1, \mathcal{V})$, for $t \in [0, 1]$. From the compacity of $\mathcal{V} \cap \mathbb{R}^n$, the smoothness of $\mathcal{V}$ and $K(\Pi_1, \mathcal{V})$ and the assumption (H_2), one deduces a continuous path in $K(\Pi_2, \mathcal{V}) \cap D_{[a,b]}$ which links x' and y'. This configuration is illustrated in Figure 3.

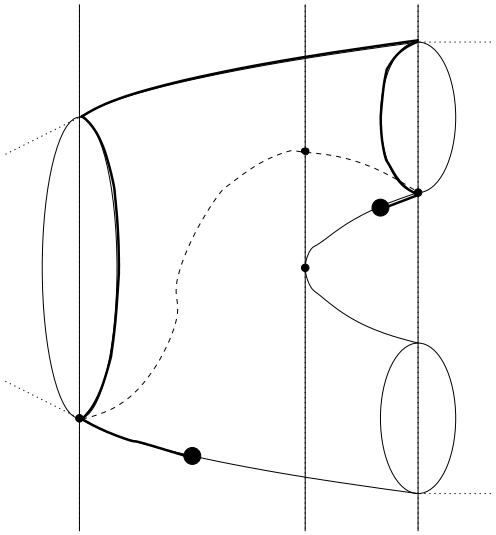


Figure 3: Local connectedness.

Suppose now that $\Pi_1(x) > v$ and $\Pi_1(y) > v$. If they are both connected to the unique critical point p such that $\Pi_1(p) = v$ by a path in $K(\Pi_2, \mathcal{V}) \cap D_{[a,b]}$ then we are done. If both of them lie on a connected component of $K(\Pi_2, \mathcal{V}) \cap D_{[a,b]}$ which does not contain p, then they can be connected by $D_{[a,b]} \cap \Pi_1^{-1}(b)$ using the properties of locally trivial fibration of Π_2 and the smoothness of $K(\Pi_1, \mathcal{V})$. The same reasoning is done when one of them is connected to p: in this case the connected component of $K(\Pi_2, \mathcal{V})$ containing p has obviously a non-empty intersection with $\Pi_1^{-1}(]v, b])$.

The situation where $\Pi_1(x) < v$ and $\Pi_1(y) < v$ is symmetric to the above one.

At last, one has to deal with the situation where $\Pi_1(x) = v$ and/or $\Pi_1(y) = v$. Since $K(\Pi_1, \mathcal{V})$ is smooth, they are connected to points x' and y' which are in one of the above cases.

□

Algebraic representation of a curve. We focus now on how to represent algebraic curves. Given a polynomial system generating a 1-dimensional ideal, one expects to obtain a parametrization of the curve with coefficients in $\mathbb{Q}(u)$ where u is a parameter.

$$\begin{cases} X_n & = & \frac{q_n(u,T)}{q_0(u,T)} \\ & \vdots & \\ X_1 & = & \frac{q_1(u,T)}{q_0(u,T)} \\ q(u,T) & = & 0 \end{cases}$$

Such a representation can be valid only outside a finite set of values of the parameter u. Indeed, for almost all specialization e of the parameter u one should retrieve a parametriza-

tion of the zero-dimensional set of points which is the intersection of the curve with the hyperplane $u = e$. The variable T encodes the separating element.

From such a representation, one can compute the number of connected components of the curve (see [4]). From several representations of that kind one can deduce a single one.

Such a parametrization can be computed from a Gröbner basis using [7] or [8] of the input polynomial system and several computations of Rational Univariate Representations (see [17]) using interpolation techniques. Other techniques based on the representation of polynomials by straight-line programs going back to [9, 10, 11] can be used, and more particularly the algorithm of geometric resolution (see [12] and [16]).

Algorithm 2.2. RoadSubRoutine:

- *Input*: A set of polynomials $(f_1, \ldots, f_s)$ in $\mathbb{Q}[X_1, \ldots, X_n]$ generating an equidimensional radical ideal of dimension d and such that $\mathcal{V}(f_1, \ldots, f_s)$ is smooth and compact.
- *Output*: A set $\mathcal{R}$ of parameterizations encoding a silhouette and a set $\mathcal{S} \subset \mathbb{Q}$ of rational numbers.

1. Compute a rational parametrization $\mathcal{R}$ encoding the critical locus of Π_2 restricted to $\mathcal{V}(f_1, \ldots, f_s)$.
2. Compute the critical values of Π_1 restricted to the curve encoded by $\mathcal{R}$.
3. Construct a set $\mathcal{S}$ of rational numbers such that there is exactly one element of $\mathcal{S}$ between each critical value of Π_1 restricted to the curve encoded by $\mathcal{R}$.
4. Return $\mathcal{R}$ and $\mathcal{S}$.

Remark 2.3. The critical locus of Π_2 restricted to $\mathcal{V}(f_1, \ldots, f_s)$ can be obtained by the vanishing of suitable minors of $\mathrm{Jac}(f_1, \ldots, f_s)$ or by using Lagrange's system. In the latter case, one has to compute a rational parametrization of the critical locus after eliminating Lagrange's multipliers.

The critical values of Π_1 restricted to $K(\Pi_2, \mathcal{V}(f_1, \ldots, f_s))$ can be obtained by computing the discriminant of the polynomial $q(u, T)$ in the parametrization $\mathcal{R}$ if the separating element can be chosen equaled to X_1. Since one works with generic coordinates, this is the case.

Algorithm 2.4. CompactRoadMap:

- *Input*: A set of polynomials $(f_1, \ldots, f_s)$ in $\mathbb{Q}[X_1, \ldots, X_n]$ generating an equidimensional radical ideal of dimension d and such that $\mathcal{V}(f_1, \ldots, f_s)$ is smooth and compact.
- *Output*: A set of parameterizations encoding a roadmap in $\mathcal{V}(f_1, \ldots, f_s) \cap \mathbb{R}^n$.

1. $\mathcal{L} := []$; $F := [f_1, \ldots, f_s]$
2. if $n = 2$ return F
3. $\mathcal{R}, \mathcal{S} :=$ `RoadSubRoutine`$([f_1, \ldots, f_s])$

4. $\mathcal{L} := \mathcal{R} \cup \mathcal{L}$
5. $\mathcal{L} := \mathcal{L} \cup (\cup_{s \in \mathcal{S}} \texttt{CompactRoadMap}(\texttt{Evaluate}(X_1 = s, F)))$
6. return $\mathcal{L}$

Remark 2.5. This is an approximative description of the algorithm: specializations of the initial variables should be memorized at each recursive call to recover a description of the roadmap in $\mathbb{R}^n$.

Algorithm 2.6. RoadMap:

- *Input*: A set of polynomials $(f_1, \ldots, f_s)$ in $\mathbb{Q}[X_1, \ldots, X_n]$ generating an equidimensional radical ideal of dimension d and such that $\mathcal{V}(f_1, \ldots, f_s)$ is smooth.
- *Output*: A set of parameterizations encoding a roadmap in $\mathcal{V}(f_1, \ldots, f_s) \cap \mathbb{R}^n$.

1. Compute the critical values of the square of the euclidean distance to a generic point $A = (a_1, \ldots, a_n) \in \mathbb{Q}^n$ restricted to $\mathcal{V}$.
2. Choose a rational R greater than the maximum of these critical values.
3. Return the result provided by `CompactRoadMap` with input $f_1, \ldots, f_s, (X_1 - a_1)^2 + \ldots + (X_n - a_n) + X_{n+1}^2 - R$ in $\mathbb{Q}[X_1, \ldots, X_{n+1}]$ after having performed a randomly chosen linear change of variables $\mathbf{A} \in GL_{n+1}(\mathbb{Q})$ on these polynomials.

Remark 2.7. Note that, obviously, if the input variety is already known to have a compact real counterpart, the only useful step in `RoadMap` is the step of linear change of variables. The step of intersecting the variety with the hyperball $(X_1 - a_1)^2 + \ldots + (X_n - a_n) + X_{n+1}^2 - R$ should be avoided as soon as it is possible since it multiplies by 2 the degree of the studied variety by `CompactRoadMap`.

Proof of correctness of the Algorithm. From Theorem 1.1, it is sufficient to prove that `CompactRoadmap` returns the correct result. The proof of correctness is done by induction on the dimension of the studied variety, following [4]. In the case of 1-dimensional variety, we are done. Consider now the general case. We partition the X_1-axis by intervals containing a single critical value of Π_1 restricted to $K(\Pi_2, \mathcal{V})$. Then, we make use of Proposition 2.1 to construct the roadmap by eventually passing through the slices if necessary following the proof of [4, Chapter 15, Lemma 15.8]. Since each slice has a dimension lower than the one of the studied variety, the induction hypothesis can be applied and each recursive call to `CompactRoadmap` returns the correct result. □

3 Complexity Estimates and Implementation

We estimate the complexity of our algorithm in the case where the input polynomial system is a complete intersection, i.e. $s = n - d$. Denote by $\mathtt{C}(n, d, D)$ the cost of the procedure `RoadSubRoutine` and by $\mathtt{H}(n, d, D)$ the number of recursive call to `CompactRoadmap` at step 5 of this procedure and by $\mathtt{T}(n, d, D)$ the total cost of our algorithm where:

- n denotes the number of variables,
- d denotes the dimension of the studied variety,
- D denotes the degree of the input polynomial system.

The cost of our algorithm is:

$$\mathtt{T}(n, d, D) = \mathtt{C}(n, d, D) + \mathtt{H}(n, d, D)\mathtt{T}(n - 1, d - 1, D)$$

We make use of the results of [19] which shows how to use Lagrange's system in conjunction with Lecerf's results [16] to improve the complexity of computing critical points and [16] which bounds the complexity of computing a lifted curve as a parametrized geometric resolution of 1-dimensional variety defined by polynomials of degree D by $D^{\mathcal{O}(n)}$. One deduces that computing a rational parametrization encoding a curve has a cost which is $D^{\mathcal{O}(n)}$ also. Using Remark 2.3, one deduces that $\mathtt{C}(n, d, D) = D^{\mathcal{O}(n)}$.

Remark now that in worst cases, $H(n, d, D)$ is also bounded by $D^{\mathcal{O}(n)}$. Since at each step, the dimension decreases, one deduces that our algorithm has a complexity within $D^{\mathcal{O}(nd)}$ arithmetic operations in $\mathbb{Q}$ which improves the one of [3]. Nevertheless, note that while the algorithm of [3] is deterministic, ours is probabilistic: it relies on assumptions on the genericity of the initial linear change of variables (nevertheless note that the assumptions (H_1) and (H_2) can be checked using tools such as Gröbner bases) and our complexity analysis relies on the use of an algorithm which is itsself probabilistic. Such comparisons are only relevant since the first choice of generic projections is, *in practice*, correct; this partially explains why our implementations are efficient. Providing a deterministic algorithm to compute roadmaps in d-dimensional algebraic varieties having a complexity within $D^{\mathcal{O}(nd)}$ arithmetic operations seems to remain an open problem.

Nevertheless, remark that the number of recursive call $H(n, d, D)$ is strongly related to the geometry of $\mathcal{V} \cap \mathbb{R}^n$ and is hopefully often less than the Bézout bound. If h denotes the maximum of $\{H(n - i, d - i, D), i = 0, \ldots, d - 1\}$ on an instance, our complexity, expressed in terms of h becomes $h^d D^{\mathcal{O}(n)}$ which is more realistic in regard of the practical behavior of our algorithm.

Moreover, the results of Lecerf allow to precise the complexity constant which is here as an exponent. This will be done in a longer version of a paper presenting this work.

A very preliminary implementation of `CompactRoadmap` has been done in `Maple` using Gröbner bases (`FGb` software, written in `C` by J.C. Faugère) and Rational Univariate Representation (`RS` written in `C` by F. Rouillier) and the interface of these softwares with `Maple`. For the moment the output is a set of 1-dimensional Gröbner bases. Up to our knowledge,

it is the first implementation computing roadmaps of real algebraic sets which is not based on Cylindrical Algebraic Decomposition. It already allows to compute roadmaps in smooth algebraic sets lying in $\mathbb{C}^6$ of dimension 5 which seems to be out of the domain reachable by Cylindrical Algebraic Decomposition. Timings obviously show the practical impact of the quantity h. This implementation will be integrated in the Maple Library `RAGLib` which is available at `http://www-calfor.lip6.fr/~safey`.

References

[1] P. Aubry, F. Rouillier, and M. Safey El Din. Real solving for positive dimensional systems. *Journal of Symbolic Computation*, 34(6):543–560, 2002.

[2] B. Bank, M. Giusti, J. Heintz, L.-M. Pardo, Generalized polar varieties: Geometry and algorithms, Journal of Complexity, 21 (4) 377-412, 2005.

[3] S. Basu, R. Pollack, M.-F. Roy, Computing roadmaps of semi-algebraic sets on a variety, Journal of the AMS, 3 (1) 55-82, 1999.

[4] S. Basu, R. Pollack, M.-F. Roy, Algorithms in real algebraic geometry, Springer-Verlag, 2003.

[5] J. Canny, The complexity of robot motion planning, MIT Press, 1987.

[6] J. Canny, Computing roadmaps in general semi-algebraic sets, The computer Journal, 1993.

[7] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4).-. *Journal of Pure and Applied Algebra*, 139(1–3):61–88, 1999.

[8] J.-C. Faugère. A new efficient algorithm for computing Gröbner without reduction to zero (F5). In *Proceedings of ISSAC 2002*, pages 75 – 83. ACM Press, 2002.

[9] M. Giusti, K. Hägele, J. Heintz, J.-E Morais, J.-L. Montaña, and L.-M. Pardo. Lower bounds for Diophantine approximation. In *Proceedings of MEGA'96*, number 117, 118 in Journal of Pure and Applied Algebra, pages 277–317, 1997.

[10] M. Giusti, J. Heintz, J.-E. Morais, J. Morgenstern, and L.-M. Pardo. Straight-line programs in geometric elimination theory. *Journal of Pure and Applied Algebra*, 124:101–146, 1998.

[11] M. Giusti, J. Heintz, J.-E. Morais, and L.-M. Pardo. When polynomial equation systems can be solved fast? In *Proceedings of AAECC-11*, volume 948 of *LNCS*, pages 205–231. Springer, 1995.

[12] M. Giusti, G. Lecerf, and B. Salvy. A Gröbner free alternative for polynomial system solving. *Journal of Complexity*, 17(1):154–211, 2001.

[13] L. Gournay and J.-J. Risler. Construction of roadmaps in semi-algebraic sets. *Appl. Alg. Eng. Comm. Comp.*, 4(4):239–252, 1993.

[14] D. Grigoriev, N. Vorobjov Counting connected component of a semi-algebraic set in subexponential time. *Comput. Complexity*, 2(2):133–186, 1992.

[15] J. Heintz, M.-F. Roy, P. Solerno. Single exponential path finding in semi-algebraic sets II: The general case. *Algebraic geometry and its applications, collections of papers from Abhyankar's 60-th birthday conference*, Purdue University, West-Lafayette, 1994.

[16] G. Lecerf. Computing the equidimensional decomposition of an algebraic closed set by means of lifting fibers. *Journal of Complexity*, 19(4):564–596, 2003.

[17] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Journal of Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.

[18] M. Safey El Din, É. Schost, Polar varieties and computation of one point in each connected component of a smooth real algebraic set, Proceedings of ISSAC 2003, 2003.

[19] M. Safey El Din, P. Trébuchet, Strong bi-homogeneous Bézout theorem and its use in effective real algebraic geometry, in preparation, 2005.

[20] J. Schwarz, M. Sharir, On the piano mover's problem II: General techniques for computing topological properties of real algebraic manifolds, Adv. Appl. Math., 4 298-351, 1983.

Marc MEZZAROBBA
Ecole Normale Supérieure de Paris
Marc.Mezzarobba@ens.fr
http://www.eleves.ens.fr/home/mezzarob/

Mohab SAFEY EL DIN
LIP6 CalFor, INRIA/LIP6 SALSA Project
University Pierre et Marie Curie
Mohab.Safey@lip6.fr
http://www-calfor.lip6.fr/~safey

Notation Selection in Mathematical Computing Environments

Elena Smirnova Stephen M. Watt

Abstract

We examine the problem of notation selection in mathematical computing environments. Users of mathematical software may require different notations for the same expression in a variety of settings. How this can be managed in a general way is the subject of this paper. We describe a software tool that can be configured to allow mathematical packages to provide output according to specified notation preferences. We explore how the choice of a set of notations can be used to disambiguate mathematical input and output in a variety of settings, including mathematical handwriting recognition, mathematical knowledge management and computer algebra systems.

1 Introduction

One of the problems for natural use of mathematical expressions with a computer is the absence of unique relationship between the syntactic presentation of mathematical objects and their semantic meaning. On one hand, the *same notation* can denote several *different mathematical objects*. For example, $\tan^{-1}$ in $\tan^2(x) + \tan^{-1}(x)$ may mean "cotangent" or "arctangent". On the other hand the *same mathematical object* can be written using *several different notations*: for instance, inner product has several notations, including: $\langle p, q \rangle$, $p \cdot q$ and $p \mathbin{\mathsf{L}} q$.

Notational ambiguity leads to possible misinterpretation of mathematical content not only by human reader, but also by automated tools, such as computer algebra systems, theorem provers, mathematical data format converters and pen-based applications. Misunderstanding of a mathematical notation will lead to unexpected or incorrect results, which in the case of automated environments can be hard to track down. All of this suggests the utility of additional tools to allow selection of default and preferred notations for use within mathematical software environments and in human-computer interfaces. This paper presents our approach to this problem.

Previous work on notation selection in mathematical context were reported in [5], [6], [11] and [17]. In this paper we elaborate on an approach to mathematical notation selection based on the idea of using meta-stylesheets for the conversion of mathematical documents [14]. We present an implementation in the form of a Notation Selection Tool for the XML-based mathematical data formats MathML[2] and OpenMath[3].

The rest of the paper is organized as follows: Section 2 gives some examples of common notational ambiguities and their origins. Section 3 describes our implementation of a Notation Selection Tool. In Section 4 we go on to consider domains of application for the notation selection technique and suggest how mathematical notation can be used for mathematical context management within various environments. The final section presents our conclusions.

2 Mathematical Notation

2.1 Types of ambiguity and their origin

When we deal with written mathematics in various domains, there are often situations where *one notation* is used to represent completely *different mathematical ideas*. For example the expression u' can mean "derivative", "minute", "logical not", "group inverse", "transformation of u", or have any one of several other interpretations. Usually the meaning can be determined from the context, but when several domains of mathematical science are used together, the meaning can be unclear and alternative notations may be required.

Equally well, *one mathematical idea* often will have *several different notations*. For example, the partial derivatives of a function may be presented in any of several ways, including: f_x, f'_x, $\partial_x f$, $\bigtriangledown_x f$, $\frac{\partial f}{\partial x}$ or $\mathrm{D}_x f$. Furthermore, it is not only the different symbols and structures that may cause ambiguity: in many cases the spatial layout of an expression is vital. For example, in various settings $C(n,k)$, ${}_nC^k$, ${}^nC_k$, ${}_nC_k$, C^k_n and C^n_k will be used for the same value – the binomial coefficient "n choose k", $\binom{n}{k}$. While the first four notations can be easily understood, the pair C^k_n and C^n_k present a serious ambiguity.

Multiple notations for the same mathematical concept may exist for a number of reasons:

- The *mathematical context* often affects the usual appearance of a formula, e.g. the fractional power of an expression can be denoted as $\sqrt[n]{f}$ or as $f^{1/n}$.
- The *area of application* may imply a default notation, for example i for $\sqrt{-1}$ in complex analysis vs. j for the same value in electrical engineering. Likewise mathematicians commonly write integration as $\int f(t)\mathrm{d}t$, while in physics the notation $\int \mathrm{d}t f(t)$ is often preferred.
- Sometimes *national and cultural conventions* apply. For example, the tangent function is presented by "tan" in Western Europe and North America but by "tg" in Eastern Europe and China. The open interval, usually denoted as (a,b) in North America, would be written as $]a,b[$ in France and some other countries.
- The *historical period* also leads to different notations. For instance, the older use of lines for grouping, *e.g.* $3\overline{a+b}$, versus the modern use of parentheses, *e.g.* $3(a+b)$.
- The *level of mathematical sophistication* may influence the preferred representation of expressions. For example, $a \div b$ and $b\overline{)\,a}$, are usually used in elementary school arithmetic, while a/b and $\frac{a}{b}$ are used at higher levels.

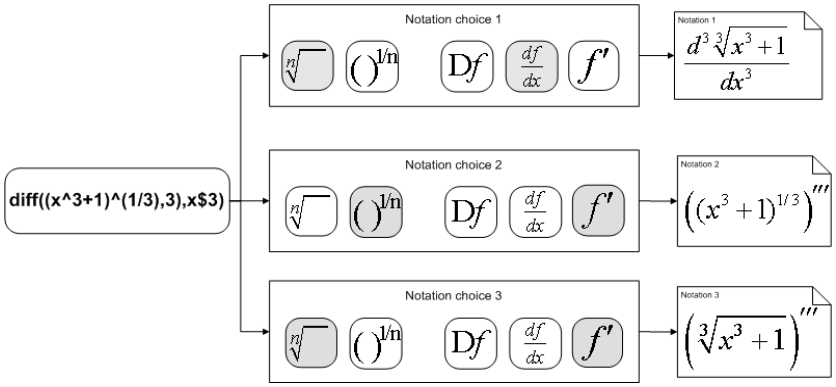


Figure 1: Rendering of mathematical objects in different notations

2.2 Notation decisions

To render a document that contains the *semantics* of mathematical objects, software tools typically use some specific set of given notations. This is the approach typically used in stylesheets for displaying conceptually-oriented Content MathML and OpenMath. Default notions are also used by mathematical software in a "pretty-printing" of computation results and in exporting output to presentational formats, such as LaTeX or Presentation MathML.

While this approach is often satisfactory, users from different scientific and cultural backgrounds may find the choice notation inconvenient. For example, for rendering an expression for "the third derivative of the cube root of the expression x cubed plus one," several notations for each operation can be used (see Figure 1). Depending on the choices for these notations and their combination, the displayed formula may either look crisp and clear or appear to be crowded and unaesthetic. In the worst case, a human reader or software package that is not familiar with a notation may not be able to determine the meaning of an expression. Unless carefully designed and maintained, software that uses fixed default notations can easily use the same notation to render different mathematical concepts. This introduces ambiguities for any later use of the output by human readers or other mathematical software.

Software with custom rendering rules can be used to avoid these ambiguities. Let us consider the ambiguous use of parentheses in the notations $\binom{m}{n}$, (a, b) and $\left(\frac{n}{p}\right)$. The first could be interpreted as a binomial coefficient or column vector. The second could be an ordered pair, an inner product, a row vector, a greatest common divisor or an interval. The third could be a Legendre symbol or a parenthesized fraction. In an output where multiple uses of the same notation would occur, these could be disambiguated by specifying the use of alternatives, *e.g.* C_m^n or $\gcd(a, b)$.

The other common situation is a mathematical concept with many notations. In this case, users may wish to choose certain specific notation for input and output of mathematical expressions. In most cases, however, only one pre-defined syntax will be accepted

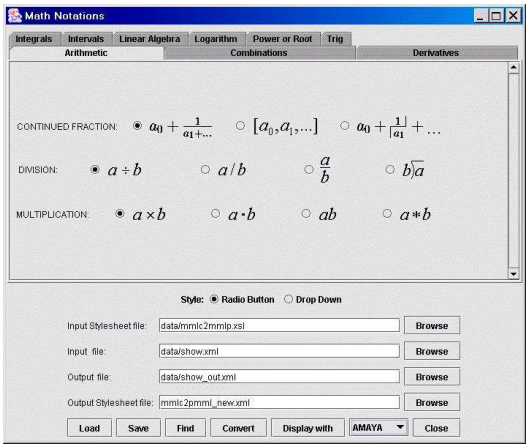


Figure 2: Stand-alone Notation Selection Tool application

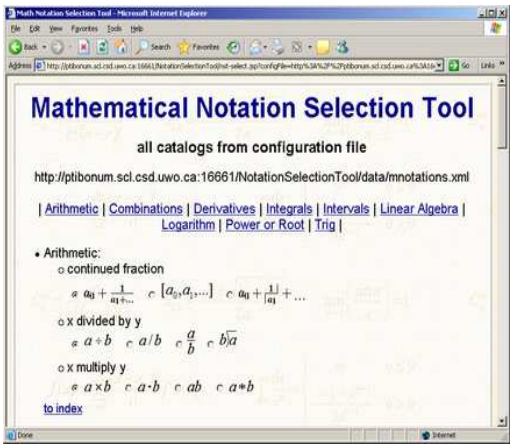


Figure 3: On-line Notation Selection Tool interface

by mathematical software packages. For instance, the notation acos(x) used by Fortran to denote "inverse of the cosine function" will not be recognized by Maple. Having the possibility to advise a mathematical software system on a set of preferred notations can serve both to disambiguate input and to produce more useful output. The collection of selected notations defines a target space of mathematical concepts. We conclude that introducing a technique to select notation for use in mathematical software would help to overcome some difficulties we have described.

3 A Notation Selection Tool

In this section we describe a software tool that allows applications to be configured to employ user-selected notations. Our *Notation Selection Tool* is designed to drive the conversion from XML-based conceptually-oriented mathematical documents into notationally-oriented formats. The Notation Selection Tool does this by generating rules to translate between mathematical expressions in different XML formats. One feature of the Notation Selection Tool is in its extensibility: an application may define the scope of mathematical concepts that tool can handle, as well as the set of alternative notations allowed for each concept.

Our implementation presents a graphical user interface that is used to generate an XSLT stylesheet. This stylesheet may then by used to convert from Content MathML to Presentation MathML. The interface allows the user to select notational conventions for a variety of concepts, organized by mathematical area. The software is deployed in two configurations: a stand-alone Java application (Figure 2) and an on-line service at `http://ptibonum.scl.csd.uwo.ca:16661/NotationSelectionTool` (Figure 3). The core of the Notation Selection Tool is written in Java. The stand-alone version uses the Swing library[9] and the on-line version is implemented with Java Server Pages [8]. The package includes a base XSLT stylesheet for Content MathML to Presentation MathML transformation [7][16], a library of images to present possible notations, and a configuration file.

3.1 Target mathematical domains

To test the feasibility of our approach, we have considered a basic set of mathematical concepts and their presentations to be handled by the notation selection tool in its default configuration. To systematize the scope of the mathematical choices, we have organized them into categories, usually corresponding to major domains of mathematics of their large subsets. We have defined the following categories: *Arithmetic*, *Combinatorics*, *Calculus*, *Linear Algebra*, *Set Theory*, *Trigonometry*. We populated each of these categories with mathematical concepts. In the category "Arithmetic" we placed operations for "multiplication", "division" and "continued fractions". The category "Calculus" originally contained operations for ordinary and partial differentiation and integration. Later as we have added more special cases for integration (limit positions, term order, *etc*), we decided to split this category into *Derivatives* and *Integrals*. We also split *Intervals* and *Logarithms* into two separate categories and created one for *Power and Root*. Each of the items within a category was then given a set of alternative notations.

Another way of systematizing mathematical domains would be to take a set of currently existing OpenMath Content Dictionaries (CDs) and create categories for each of them. Then the set of `OMS` symbols within each CD would define the content of each category. In the experimental design for our tool we decided to stay with our simplified set of mathematical operations.

3.2 The configuration file

Even though we tried to include a number of popular mathematical concepts in the initial configuration of our Notation Selection Tool, we could not guarantee that default settings will satisfy an average user. We therefore organized our software in a such a way that the end user can introduce other mathematical objects and their own notations. This is done by updating the *configuration file* that is used to initialize the Notation Selection Tool.

The configuration file contains a database of concepts, organized by category, and specifies alternative notations. We found an XML format suitable to store this file. (In particular this simplified storing template XSLT rules.) In this way the structure of the configuration file can naturally represent the classification introduced in previous section. The top-level organization of the configuration file is illustrated by the excerpt shown in Figure 4.

Each mathematical category is represented in the configuration file by the element `catalog`. Supported concepts and operations from each category have `item` elements nested inside of `catalog`. Each item is given by an OpenMath expression and is assigned a list of *notation choices* given within `choicelist` element. For instance, the operation of differentiation may be encoded as the OpenMath symbol `diff` from the Content Dictionary `weylalgebra1` and can be given the notation choices: f_x, f'_x, $\frac{\mathrm{d}f}{\mathrm{d}x}$ or $\mathrm{D}_x f$, etc.

Each notation choice, in turn, binds an example of the notation appearance (presented by a reference to an image file) to a set of converters (given by their XSLT templates). Each template defines the rules for *transformation of mathematical content to its presentation* according to the notational choice. Figure 5 provides an example of such rules to generate notations for the inverse sine function.

```
<catalog>
  <name> LINEAR ALGEBRA </name>
  <itemlist>
    <item>
      <keyword> INNER PRODUCT </keyword>
      <content>
          <!-- OpenMath encoding for mathematical concept INNER PRODUCT -->
      </content>
      <choicelist>
        <choice>
          <!-- The first notation choice for INNER PRODUCT -->
          <image src = "inn_prod1.gif"/>
          <keyvalue> 1 </keyvalue>
          <presentation>
             <converter input = "Content MathML" output="Presentation MathML">
             ... <!-- XSLT template for Content MathML to Presentation MathML -->
             </converter>

             <converter input = "OpenMath" output="LaTeX">
             ... <!-- XSLT template for OpenMath to LaTeX for this notation -->
             </converter>

             ... <!-- other possible converters -->
          </presentation>
        </choice>
        ... <!-- Other notation choices for INNER PRODUCT -->
      </choicelist>
    </item>
    ... <!-- Other items within LINEAR ALGEBRA -->
  </itemlist>
</catalog>
```

Figure 4: Configuration file excerpt

3.3 Stylesheet generation

After the user of the Notation Selection Tool selects a set of preferred notations, XSLT templates associated with each of the chosen notations are inserted into a new stylesheet. This new stylesheet combines the selected rules with the initial stylesheet supplied with our software.

The XSLT templates output by the Notation Selection Tool override the rules for the same input patterns in the base stylesheet. This is ensured by assigning higher priorities to the XSLT templates than to the corresponding templates in general stylesheet. These priorities are given by a value of the `priority` attribute appearing in each template. The generated stylesheet can then be further used within the tool or used independently for conversion of XML objects from encoding mathematical content to a presentational form (see Figure 6). We note specifically that the generated stylesheet may be used to translate to presentation form alone or it can be used in a "content-faithful" manner to produce output that retains the original semantics.

By default, the Notation Selection Tool produces a stylesheet to convert from Content MathML to Presentation MathML. The configuration file also may define templates for

```
<item>
   <title> Inverse sine of x </title>
   <keyword> ARCSINE </keyword>
   <content>
     <om:OMS cd="transc1" name="arcsin"/>
   </content>
   <choicelist>
    <!-- ******** ARCSINE CHOICE 1 ********************** -->
      <choice>
        <image src = "arcsin1.gif"/>
        <keyvalue> 1 </keyvalue>
        <presentation input="CMathML" output = "PMathML">
           <xsl:template match = "mml:arcsin" mode = "trigonometry" priority="100">
              <msup>
                 <mo>sin</mo>
                 <mn>-1</mn>
              </msup>
          </xsl:template>
        </presentation>
        <presentation input="OpenMath" output = "LaTeX">
           <xsl:template match = "om:OMS[@cd='transc1' and @name='arcsin']" priority="100">
              \sin^{-1}
          </xsl:template>
        </presentation>
      </choice>
    <!-- ******** ARCSINE CHOICE 2 ********************** -->
      <choice>
        <image src = "arcsin2.gif"/>
        <keyvalue> 2 </keyvalue>
        <presentation input="CMathML" output = "PMathML">
           <xsl:template match = "mml:arcsin" mode = "trigonometry" priority="100">
              <mo>arcsin</mo>
           </xsl:template>
        </presentation>
        <presentation input="OpenMath" output = "LaTeX">
           <xsl:template match = "om:OMS[@cd='transc1' and @name='arcsin']" priority="100">
              \mathop{arcsin}
           </xsl:template>
        </presentation>
      </choice>
    <!-- ******** ARCSINE CHOICE 3 ********************** -->
      <choice>
         <image src = "arcsin3.gif"/>
         <keyvalue> 3 </keyvalue>
         <presentation input="CMathML" output = "PMathML">
            <xsl:template match = "mml:arcsin" mode = "trigonometry" priority="100">
               <mo>asin</mo>
            </xsl:template>
         </presentation>
         <presentation input="OpenMath" output = "LaTeX">
           <xsl:template match = "om:OMS[@cd='transc1' and @name='arcsin']" priority="100">
              \asin
           </xsl:template>
         </presentation>
     </choice>
  </choicelist>
</item>
```

Figure 5: Example of rules to generate notations for the inverse sine function

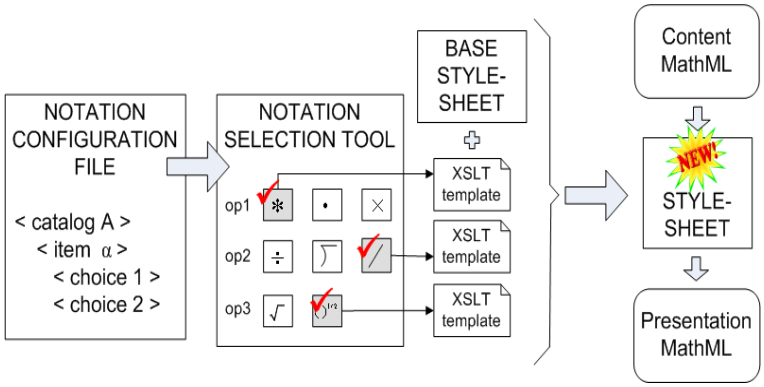


Figure 6: Notation Selection Tool in action

other types of conversion. This is indicated by the values of attributes `input` and `output`. If another type of translation is chosen, an appropriate base stylesheet must be used. The initial software kit for the Notation Selection Tool provides a base stylesheet and configuration file only for Content MathML to Presentation MathML translation. We address the question of supporting other types of conversion in Section 3.5

3.4 Extensible design

A common problem with software packages it that once they are released they do not allow users sufficient control over their behaviour. In particular, mathematical software packages often have many built-in underlying assumptions that permeate their handling of mathematical expressions. In contrast, the whole point of providing notation selection is to allows users flexibility. Ensuring flexibility and extensibility of our Notation Selection Tool has therefore been been one of our guiding principles.

To provide the desired flexibility we have used the idea of an editable configuration file. This allows the user to introduce new notations for existing math concepts by adding to this file. In particular, the set of catalogs and their mathematical content may be changed.

In the same way, new mathematical concepts can be created in existing settings. Doing this entails introducing notational choices, backed by stylesheet tools, to act as targets of those choices. For example, binomial coefficients and continued fractions are defined neither in Content MathML nor in Presentation MathML. They can be presented, however, through the use of annotated `<csymbol>` elements or by stylesheet templates for newly-defined elements. Annotated `<csymbol>` elements use a `definitionURL` attribute to reference some agreed-upon definition. (There is not necessarily any data *at* the URL location.)

```
<apply>
   <csymbol definitionURL="http://orcca.on.ca/MathML/newelement.html#binom">
   <ci> n </ci>
   <ci> m </ci>
</apply>
```

To introduce new elements in Presentation MathML we can define new XML structures as extensions of MathML. The elements of this extended MathML may be identified using a specific namespace. Thus, different notations for binomial coefficients in Presentation MathML can be defined using an XML Schema or DTD:

```
<!ELEMENT mmlx:binom( math:mi, math:mi)>
<!ELEMENT mmlx:choose( math:mi, math:mi)>
```

To define the rendering for these new elements, one can define the following XSLT transformation rules:

```
<xsl:template match = "apply/mmlx:binom[position()=1][count(child::*)=2]">
  <msupsub>
    <mfrac thickness="0ex">
      <mo> C </mo>
      <xsl:for-each select = 'mmlx:binom/child::*'>
         <xsl:copy-of select='.'/>
      </xsl:for-each>
    </mfrac>
  </msupsub>
</xsl:template>

<xsl:template match = "apply/mmlx:choose[position()=1][count(child::*)=2]">
  <mfenced>
    <mfrac thickness="0ex">
      <xsl:for-each select = 'mmlx:binom/child::*'>
        <xsl:copy-of select='.'/>
      </xsl:for-each>
    </mfrac>
  </mfenced>
</xsl:template>
```

The first template provides the extended Presentation MathML element `mmlx:binom` with the notation C_{arg2}^{arg1}. The second generates the presentation $\binom{arg1}{arg2}$ for the element `mmlx:choose`.

The same approach allows one to set preferred renderings for OpenMath CDs with Presentation MathML, as not every OpenMath symbol has corresponding elements in Presentation MathML. We can apply this technique to introduce new elements to Presentation MathML. The new elements may then be used in XSLT templates within the notation configuration file. In this way, OpenMath symbols can be mapped directly to the set of extended MathML elements. This is beneficial in two ways: First it makes the correspondence between OpenMath entities and their presentation more natural. Second this ease the creation of XSLT templates for the configuration file and allows a more compact format for the configuration file itself.

3.5 Notation selection in combination with additional transformations

So far, we have concentrated on the use of our Notation Selection Tool to build MathML transformation stylesheets. The tool is designed, however, to drive transformations between a wider range of data formats, as shown in Figure 7. The common characteristics of these

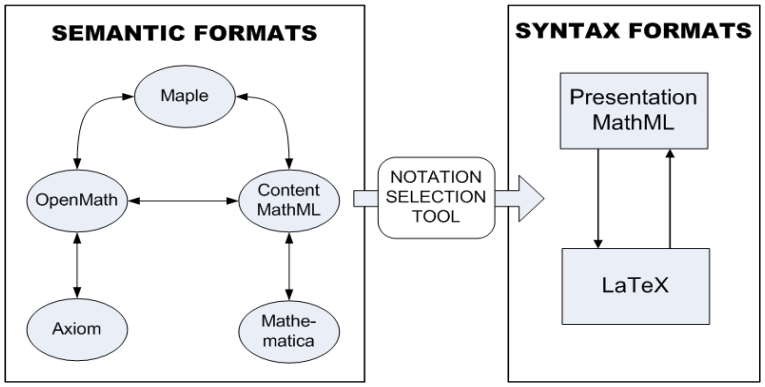


Figure 7: Mathematical data format translations implemented

conversions is that they typically take objects from high-level semantic views to lower-level renderings.

One way to enable these transformations is to introduce new entries in the configuration file. This can be done by placing additional XSLT templates within the `<presentation>` element and specifying values of the `input` and `output` attributes to indicate the type of translation. A second approach is to keep the variety of XSLT templates in the configuration file to the minimum possible, *i.e.* to support only one type of conversion, for instance Content MathML to Presentation MathML. For the other formats, external conversion tools can be used. Our group at the Ontario Research Centre for Computer Algebra has developed various data format translation techniques supporting conversion[1], shown in Figure 7.

OpenMath ↔ Content MathML The conversion between these two XML-based formats is performed natively by XSLT stylesheet transformations. We have designed the stylesheets for a 2-step transformation in both directions. Conversion from OpenMath to Content MathML may either elect to generate the built-in MathML operations, when appropriate, or to give everything in a general format using `csymbol`s. In this direction our converter supports arbitrary OpenMath CDs. Conversion in the other direction, from Content MathML to OpenMath, understands the standard OpenMath CDs.

OpenMath ↔ Maple The translator between Maple objects and OpenMath expressions is organized as an engine driven by a set of mapping files. These files describe the correspondence between patterns of mathematical structures represented by both formats. The scope of the mathematical content that the converter can handle is determined by the set of mapping files it has loaded. This allows us to configure the converter vocabulary, which

[1]The conversions {Maple, Mathematica} → Content MathML and Axiom → OpenMath are internally supported in the corresponding systems. The conversions {Maple,Mathematica} → Presentation MathML and {Maple,Mathematica,Axiom} → LaTeX are also supported internally, but these conversions use only default notation, defined by the systems.

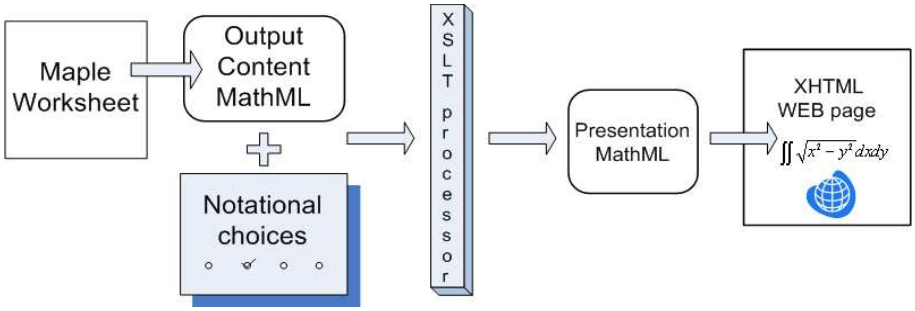


Figure 8: Using notation selection to produce web pages with mathematical content

is necessary to support OpenMath's extensible nature. It is convenient for every mapping file to correspond to one OpenMath Content Dictionary. It is used to drive both directions of the translation. The translator engine itself is implemented in Maple, which makes the conversion tool native to the environment of the computer algebra system, while remaining configurable by the user.

Presentation MathML ↔ LaTeX Even though the most common approach to convert from MathML to TeX is, again, to use XSLT stylesheets, we chose *not* to use this method. The reason is that we wanted our converter to be symmetric to our TeX to MathML converter and for both of them to be configured by one set of bidirectional mappings. Since TeX is not an XML-based format, we could not use XSLT to translate from TeX to MathML. Therefore, we have developed two Java packages, one for each direction of conversion to and from TeX. Both programs use the *same* mapping file to set the correspondence between MathML and LaTeX syntactic patterns. This mapping file is similar to the configuration file used by the Notation Selection Tool and can be edited by the users of the converter. This approach allows flexibility and extensibility of the tools for these conversions. The other benefit of using mapping files is preserving high-level semantics in transforming between TeX macros and MathML extension elements. The on-line version of the LaTeX ↔ MathML translator is available at `http://www.orcca.on.ca/MathML/texmml`. More details on the MathML to LaTeX converter are given in [21].

4 Applications of Notation Selection

One of the most common applications for the Notation Selection Tool is support for multiple formats in mathematical documents. One can imagine a likely scenario in which a user may wish to export the computation results from a computer algebra system and to place them on a web page within lecture notes or on-line publication. The rendering of mathematical content then may be specified by a choice of notation set trough the Notation Selection Tool (see Figure 8).

Another area of application for our Notation Selection Tool is that of mathematical education, where students require a high degree of notational consistency within a syllabus. Notation selection facilities can allow an instructor to re-use material with different notational conventions from one course to another. In distance learning, students might prefer to see mathematical expressions in the format of their locality, so our tool could be used to choose these preferences.

We describe below a few of our on-going investigations which involve our Notation Selection Tool.

4.1 From syntax to semantics

We have discussed how, at a syntactic level, mathematical notation can be highly ambiguous. This may not be apparent to most mathematical readers, as they will be using a great deal of semantic contextual information and moreover any particular document will make use of only a few areas of mathematics. We suggest, therefore, that the general semantic attribution of pure syntactic mathematical expressions is a suitable well-defined problem in the area of artificial intelligence.

From the point of view of mathematical software, however, we do not necessarily require handling of general expressions in every possible context. It may be perfectly acceptable to require the user, or application, to specify explicitly the mathematical context in which the expressions occur. In this case, with the mathematical domain suitably narrowed, semantic attribution of expressions becomes much more tractable.

We anticipate that the use of tools, such as our Notation Selection Tool, will be useful in this setting. By selecting a set of notational preferences, the user defines a space of admissible expressions together with their interpretation. Clearly, not all possible combinations of selected notations will lead to unique interpretation of all expressions. However, once a selection of preferences is made, it is possible to determine in advance the areas where ambiguities will arise (see Figure 9).

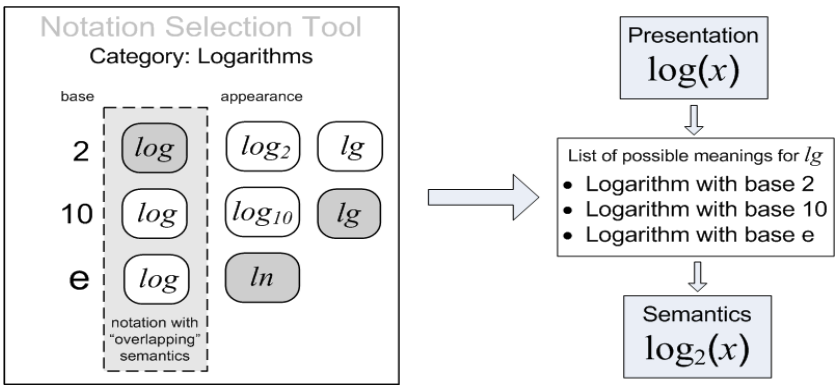


Figure 9: Notation selection in syntax disambiguation

4.2 Notation selection in mathematical handwriting recognition

Input to mathematical software comes in the form of expressions in various parsed linear syntaxes, expressions built using interactive expression editors, in MathML or TEX. We are currently engaged in a project that attempts to add handwritten mathematics as an input method[18]. Handwritten input presents an interesting and difficult case, because it relies both on accurate single character recognition within a large set of mathematical symbols and on structural analysis of 2-dimensional layout. We see an opportunity for the Notation Selection Tool to assist in this process, both for handwritten input to computer algebra systems (see Figure 10) and other applications (Figure 11).

As a high-level design objective, our expression recognizer must support various notations in handwritten input. This leads to problems, however, when notations are ambiguous or very similar. Attempting to disambiguate between many such choices would be both time consuming and reduce recognizer accuracy. To narrow down the set of expression models without forcing the user's choice of notation, we can ask the user to use our Notation Selection Tool to specify preferred notations.

Eliminating un-used notations has a first benefit of reducing the set of candidates in single character analysis. For example, if a user has selected $\sim$ instead of $\propto$ for proportionality, then $\propto$ can be eliminated from character candidate lists. This will lead to more accurate recognition of similar characters, such as α and ∞.

Once character candidates have been generated and ranked, it is necessary to select the most likely choice while taking into account expression context analysis. Notation selection can be used here to activate or deactivate various rules in the structural recognition process. For example, if French-style interval notation is selected, then parentheses and brackets do not have to match.

At present, our recognizer uses a combination of single character analysis and character prediction. The prediction is based on writing order Markov chains compiled from the analysis of some 20,000 mathematical documents. The source for these mathematical documents, however, is in TEX form and there is no distinction made in the analysis between different meanings for the same notation [19][20]. We have not yet explored whether notation selection applied to the document database would lead to more better prediction. A second use of notation selection that we have not yet explored for mathematical handwriting recognition is the use of correlated notations. For example, if the notation $\int \mathrm{d}t f(t)$ is selected for integration, then it would be reasonable to assign higher likelihoods to other notations from physics.

4.3 Notation selection in mathematical knowledge management

One of the areas in mathematical knowledge management is to organize or classify entries in mathematical knowledge bases. Such categories are well defined and supported by a number of systems and databases including Mizar [13], MBase [10], the NIST Digital Library of Mathematical Functions [12], OpenMath Content Dictionaries [3], to name a few. The work [15] is also interesting in this regard.

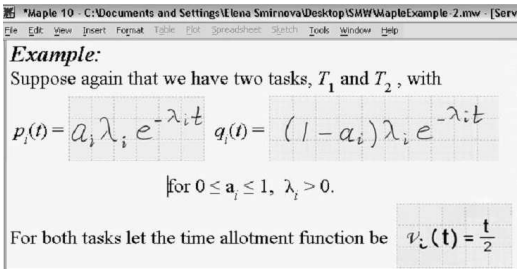


Figure 10: Pen-Enabled input in Maple

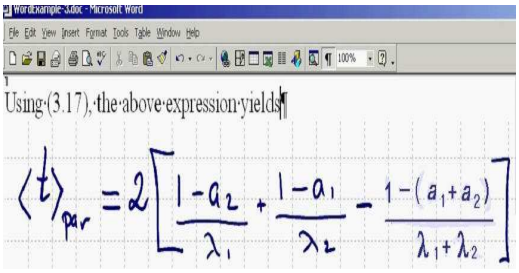


Figure 11: Pen-Enabled input in Word

A goal with these categorizing tools is to avoid storing duplicated information. Each knowledge base will choose its representations for particular functions, and organize information about them accordingly. For example, if information about the hyperbolic sine function uses the function $\sinh(z)$ then users from Eastern Europe will find nothing about $\mathrm{sh}(z)$ because there is no such entry.

One possible solution is to include multiple function names in the database, but this is fraught with problems. If, for example, rules are modified for $\arcsin(z)$ but not for $\sin^{-1}(z)$, is this an intentional subtle distinction between inverse functions with different branch cuts or is it a bug? Should $\arcsin(x) - \sin^{-1}(z)$ simplify to zero? It is clearly preferable to use a notation selection tool as a component to the front end to such mathematical knowledge bases.

4.4 Notation selection in computer algebra systems

Most of the issues we have raised so far is directly applicable to the input and output of computer algebra systems. The ability adapt the interpretation of input forms and to select the possible forms of output has obvious value in making these systems more accessible to different communities.

There are certain practical considerations, however, that impact the successful adoption of this strategy in current systems. The most widely used mathematical software packages are designed with the input and output systems separate from one another. In this situation, to support proper notation selection, it is necessary to apply transformations to *both* the input and output expressions. Difficulties arise when the input and output transformations are inconsistent. This makes it difficult to effectively modify most existing computer algebra systems to support notation selection coherently: both the input and output systems are typically sizable, complex software subsystems, and obtaining consistency between them is challenging. We have experienced this situation with our suite of transformation tools to convert between various mathematical formats, and in particular converting between TEX and MathML[2]. In this situation we have found a useful approach is to specify a set of bi-directional transformation rules, used independently by the translators for both directions.

Beyond the question of consistent handling of input and output notation selection, computer algebra systems present an additional problem: They generate new mathematical expressions as part of their operation and these expressions may contain functions or other symbols not anticipated by the user. For example, in the solution of differential equations a wide variety of special functions may be generated, some of which might be unknown to the user of the system. It is unreasonable to disallow the user to employ notations that denote standard functions. There are so many that *could* occur, but only a few actually *will* in any particular setting. On the other hand, always using non-standard names for standard functions (e.g. `BesselJ[0](z)` instead of $J_0(z)$) leads to bulky expressions that may not be used directly in other settings. It is therefore necessary to properly manage the interactions between notations deliberately set by the user and names that may occur in generated expressions.

5 Conclusion

We have observed that a wide range of software, including document processors and computer algebra systems confuse *mathematics* with *notation*. By forcing too early the choice of notation, flexibility is lost and work is restricted to a too narrow context. We have shown how software tools can be used to translate meaningful mathematical constructs both to and from a wide range of notations. This can be used to allow mathematical documents and computational worksheets to be deployed in a wide range of settings, with notation customized by country, mathematical field, level of sophistication, or other criterion.

We have described the rationale, design and implementation of a Notation Selection Tool that allows the interactive construction of stylesheets to convert between mathematical formats. The current state of the art is presently implemented for transforming mathematical data encoded in Content MathML to Presentation MathML. When used in conjunction with additional conversion tools the following translations are also available: {Content MathML, OpenMath, Maple, Axiom, Mathematica} → {Presentation MathML, LaTeX}.

We have discussed a number of settings where notation selection tools can be useful in defining a restricted domain of discourse. This disambiguation can be useful in settings that require some semantic treatment of a wide range of mathematical subjects. In particularly, we have discussed notation selection in the context of computer algebra, in improving accuracy in pen-based mathematical interfaces and in mathematical knowledge management applications.

We see an interesting future project in incorporating a flexible Notation Selection Tool in major computer algebra systems, such as Maple. For maximum utility, this must work bi-directionally: to select notations to be used for both input and output. Mathematical structure editors can allow us to avoid the technical parsing problems that would be introduced through user-defined linear input syntax.

References

[1] M. Abramowitz and A. Stegun: Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables.

[2] R. Ausbrooks at al. Mathematical Markup Language (MathML) Version 2.0 (Second Edition), World Wide Web Consortium Recommendation 21 October 2003. http://www.w3.org/TR/2003/REC-MathML2-20031021

[3] S. Buswell *et al.*, The OpenMath Standard 2.0. 2004 http://www.openmath.org/cocoon/openmath/standard/om20/index.html

[4] James Clark, World Wide Web Consortium Recommendation, XSL Transformations (XSLT), December 1999. http://www.w3.org/TR/XSLT

[5] James H. Davenport, Mathematical Knowledge Representation, Electronic Proceedings of the First International Workshop on Mathematical Knowledge Management: MKM 2001 RISC, A-4232 Schloss Hagenberg, September 24-26, 2001.

[6] Jason Harris, Advanced Notations in Mathematica, Proceedings of the International Symposium on Symbolic and Algebraic Computation, 2000.

[7] Sandy Huerter, Igor Rodionov and Stephen M. Watt, Content-Faithful Transformations for MathML, Proc. International Conference on MathML and Math on the Web, (MathML 2002), http://www.mathmlconference.org/2002, June 28-30 2002, Chicago USA.

[8] Java Server Pages, http://java.sun.com/products/jsp/

[9] Java Swing, Robert Eckstein, Marc Loy and Dave Wood, 1998, O'Reilly.

[10] M. Kohlhase, A. Franke. MBase: Representing Knowledge and Context for the Integration of Mathematical Software Systems. Journal of Symbolic Computation 23:4 (2001), pp. 365 - 402.

[11] Dicheng Liu, A Notation Selection Tools for MathML stylesheets, MSc Project University of Western Ontario, 2001

[12] D.W. Lozier, NIST Digital Library of Mathematical Functions. In Annals of Mathematics and Artifcial Intelligence, vol. 38, No. 1-3, May 2003. Eds. B. Buchberger, G. Gonnet, M. Hazewinkel. Kluwer Adacemic Publishers, ISSN 1012-2443.

[13] The Mizar System, http://mizar.uwb.edu.pl/system/

[14] W.A. Naylor and Stephen M. Watt, Meta-Stylesheets for the Conversion of Mathematical Documents into Multiple Forms, Annals of Mathematics and Artificial Intelligence, Vol. 38, pp. 3-25, 2003.

[15] Florina Piroi and Bruno Buchberger, An Environment for Building Mathematical Knowledge Libraries. Proc. Fourth International Conference on Mathematical Knowledge Management, (MKM 2005), July 15-17 2005, Bremen Germany, Springer Verlag

[16] Igor Rodionov and Stephen M. Watt, Content-Faithful Stylesheets for MathML, Ontario Research Centre for Computer Algebra, University of Western Ontario, Research Report TR-00-14, 2000.

[17] Elena Smirnova and Stephen M. Watt, An Approach to Mathematical Notation Selection, North American Mathematical Knowledge Management Workshop , (NA-MKM 2004), 2004, Phoenix Arizona.

[18] Elena Smirnova and Stephen M. Watt, A Context for Pen-Based Computing, pp. 409-422, Proc. Maple Conference 2005, July 17-21 2005, Waterloo Canada, Maplesoft.

[19] Clare M. So and Stephen M. Watt, Determining Empirical Properties of Mathematical Expression Use , Proc. Fourth International Conference on Mathematical Knowledge Management, (MKM 2005), July 15-17 2005, Bremen Germany, Springer Verlag

[20] Clare M. So, An Analysis of Mathematical Expressions Used in Practice, MSc Thesis, University of Western Ontario, 2005

[21] Stephen M. Watt, Exploiting Implicit Mathematical Semantics in Conversion between TEX and MathML, Proc. Internet Accessible Mathematical Communication, (IAMC 2002)

[22] XML specification: http://www.w3.org/XML.

Elena Smirnova
Ontario Research Centre for Computer Algebra
University of Western Ontario
elena@orcca.on.ca
www.orcca.on.ca/MathML/elena.html

Stephen M. Watt
Ontario Research Centre for Computer Algebra
University of Western Ontario
watt@orcca.on.ca
http://www.orcca.on.ca/~watt

Computing the Algebraic Counterpart of a Tropical Plane Geometric Construction

Luis Felipe Tabera

February 16, 2006

Abstract

Here, the notion (from [6]) of tropical plane geometric construction is extended to the non linear case. The possibility of lifting a construction to an algebraic setting will be studied, requiring the lifting to be compatible both with tropicalization and with the construction itself. Given a tropical geometric construction, we present a method to compute sufficient conditions for the lifting of the input elements to verify the mentioned compatibilities. These conditions are expressed as a constructible set on the residue field of the valued field the tropicalization is based on.

Introduction

An increasing interest in tropical geometry has arisen during the last years (see [3], [4]) One of the reasons for this interest lies on Mikhalkin's correspondence theorem (cf. [4]). It relates Gromov-Witten and Welschinger invariants between the tropical and algebraic setting. Thus, Mikhalkin presents an algorithm to compute all tropical curves with fixed genus and degree going through a special set of points in the plane. He also shows that this number equals the number of algebraic curves passing through a generic configuration of points. Moreover, using this correspondence, in [3], a nontrivial lower bound to the number of real algebraic nodal rational curves passing through a generic configuration of real points is provided.

The key for the success of applications like these consists in translating a given algebraic problem to a tropical context. Here, the polyhedral nature of tropical geometry usually allows an easier –and even algorithmic– manipulation of the geometric objects and hence, it facilitates to solve the algebraic problem we started with. On the other hand, it is well known that there is a loss of information when passing from the algebraic to the tropical world. For instance, not every configuration of tropical points and lines in Pappus position is the projection, over the tropical plane, of an algebraic configuration verifying the same incidence conditions (cf. [5]). These considerations yield quite naturally to consider the following question: *When does a set of tropical curves, with some specific incidence relations among them, come from a set of algebraic curves verifying the same incidence relations?*

In this extended abstract we generalize the approach of [6], where some conclusions for the linear case have already been presented. Our main result (Theorem 2.1) establishes that for configurations of tropical curves and points arising from a construction (see Definition 1.2), there is a set of algebraic elements verifying the same incidence relations if there is an algebrization of the input tropical elements in the construction (see 1.2) such that their principal coefficients (see next section) belong to a certain definable set $S \subseteq \mathbb{C}^n$, explicitly described in Section 3. The important matter is not the existence of S, but the fact that our algorithm finds a sufficiently big S in many cases. We remark that this result gives a sufficient condition for an affirmative answer to the question above. Moreover, our approach is constructive, since we give an algorithm to compute S from the construction data. Thus, it could be useful as a tool for providing constructible proofs of algebraic statements that have been obtained via tropical geometry. At this moment we will neither consider complexity issues nor other geometric characteristics of the curves, such as the genus.

1 Elementary definitions and properties in tropical geometry

As we are interested in constructive results, we choose, without loss of generality [7], to work with the field $\mathbb{K} = \overline{\mathbb{Q}(t)^{alg}}$, the algebraic closure of the field of rational functions in one variable over $\mathbb{Q}$. Every element of this field can be written as a Puiseux power series over the field of algebraic numbers $\mathbb{Q}^a$,

$$\widetilde{a} = \sum_{i=k}^{\infty} \alpha_i t^{\frac{i}{q}},\ k \in \mathbb{Z},\ q \in \mathbb{N}^*,\ \alpha_i \in \mathbb{Q}^a.$$

The order of a non zero series is $o(\widetilde{a}) = \min\{i \mid \alpha_i \neq 0\} \in \mathbb{Q}$, $o(0) = \infty$. This order is a valuation on the field $\mathbb{K}$, with valuation group $\mathbb{Q}$ and residue field $\mathbb{Q}^a$. The principal coefficient of an element $\widetilde{a} \in \mathbb{K}$ is $Pc(\widetilde{a}) = \alpha_{o(\widetilde{a})} \in \mathbb{Q}^a$. An element will usually be written as $\widetilde{a} = \alpha_{o(\widetilde{a})} t^{o(\widetilde{a})} + O(t^{o(\widetilde{a})+\epsilon})$ to emphasize the principal term of a series.

Regarding the conversion between algebraic and tropical varieties, the crucial map is the tropicalization or projection map $T : \mathbb{K} \longrightarrow \mathbb{T} = \mathbb{Q} \cup \{-\infty\}$, $T(x) = -o(x)$, $T(0) = -\infty$ (in the current literature a parallel approach can be found, which considers $o(x)$ instead of $-o(x)$ in this definition). Then, by definition, a tropical variety is the image $T(V) \subseteq (\mathbb{T}^*)^n = \mathbb{Q}^n$ of an algebraic variety $V \subseteq (\mathbb{K}^*)^n$.

Conversely, given a tropical variety C, we will talk about a *lift* of C as an algebraic variety $\widetilde{C}$ such that $T(\widetilde{C}) = C$

On the other hand, we may consider our valuation group $\mathbb{Q}$ as an idempotent semifield [5] provided with the tropical sum "$a + b$" $= \max\{a, b\}$ and product "ab" $= a + b$. Remark that operations between quotes are tropical and the symbols do not represent the standard sum and product. In this mood, a tropical polynomial f will be defined as a standard (Laurent) polynomial in the variables $x = (x_1, \ldots, x_n)$, but with tropical operations and tropical coefficients (ie. in $\mathbb{T}^* = \mathbb{Q}$):

$$f = \text{“} \sum_{i \in \Lambda \subseteq \mathbb{Z}^n} a_i x^i \text{”} = \max_{i \in \Lambda \subseteq \mathbb{Z}^n} \{a_i + ix\}$$

for a suitable set $\Lambda \subseteq \mathbb{Z}^n$, where $i = (i_1, \ldots, i_n)$ and $ix = i_1x_1 + \ldots + i_nx_n$. Let Δ be the convex hull of Λ, that is, the Newton polytope of f. Notice that, by a suitable election of some a_i's negative enough, we can have the following equality

$$\text{“} \sum_{i \in \Lambda \subseteq \mathbb{Z}^n} a_i x^i \text{ ”} = \text{“} \sum_{i \in \Delta \cap \mathbb{Z}^n} a_i x^i \text{ ”}$$

as piecewise affine continuous function $\mathbb{Q}^n \longrightarrow \mathbb{Q}$.

Remark 1.1. So, without loss of generality, in the rest of the paper we will restrict to polynomials whose support Λ equals the set of all the integer points of its Newton polytope.

With this notation, we can define the tropical hypersurface associated to the polynomial f as the set of points of $\mathbb{T}^n$ such that the piecewise continuous affine function $\max_{i \in \Delta \cap \mathbb{Z}^n}\{a_i + ix\}$ achieves its maximum in at least two different indices.

It is well known (cf. [2]) that both definitions of tropical hypersurfaces are equivalent. In fact, let us associate to a polynomial $\widetilde{f} = \sum_{i \in \Delta \cap \mathbb{Z}^n} \widetilde{a}_i x^i$ in $\mathbb{K}[x]$, the tropical polynomial $f = \text{“}\sum_{i \in \Delta \cap \mathbb{Z}^n} T(\widetilde{a}_i)x^i \text{ ”}$. Then, the tropicalization of the algebraic variety defined by $\widetilde{f}$ equals the tropical hypersurface associated to f. One advantage of this equivalence is that it allows to work with tropical curves on their own, without defining them as a projection of an algebraic curve.

A basic problem working with planar tropical curves is appropriately defining its intersection, since two different irreducible tropical curves may share an infinite number of points. This problem is avoided considering the so called stable intersection of curves [5]. Roughly speaking, this notion is built as follows: first, it is observed that given two tropical curves with an infinite intersection, this intersection becomes finite under a small generic translation of one of the curves. Second, the limit set of this collection of intersection points, as the translation becomes smaller, is a well defined finite set contained in the intersection of the original curves. This set has reasonable properties and is called the stable intersection of the two curves.

We may define analogously the notion of stable curve (of given Newton polygon Δ) containing a given number of points. For that purpose, consider the set of all curves with Newton polygon Δ passing through a configuration of $\#(\Delta \cap \mathbb{Z}^2) - 1$ points. Notice that, if the configuration of points is general enough, there will be only one such a curve. Else, it may happen that there is an infinite number of curves. In this case, we can perturb the configuration of points to a generic situation. As the perturbation tends to zero, the limit set of the curves going through the perturbed configuration, is a precise tropical curve that passes through the original configuration of points.

Another tool we will use in the presentation of the main theorem is the notion of resultant of two tropical polynomials f and g with respect to one variable x. It is analogous to the definition of resultant in the algebraic case, merely translating to the tropical framework the corresponding operations.

Finally, we present the notion of geometric construction, that appears as an essential hypothesis in Theorem 2.1.

Definition 1.2. A planar (tropical) geometric construction is an abstract procedure consisting of

- Input data: A finite set of points p_i and curves C_j with prescribed Newton polygon Δ_j.
- A finite sequence of allowable steps, namely, computing either
  - the (stable) intersection of two (already given) curves, or
  - the (stable) curve with Newton polygon Δ passing through $\#(\Delta \cap \mathbb{Z}^2) - 1$ (already given) points.
- Output: A finite set of points and curves (having some incidence relations among them).

Notice that it is intended this definition to be applicable both to the tropical and the classical case. This notion of geometric construction is general enough to be interesting on its own, since many cases of incidence relations between curves and points can be presented through a geometric construction, as in the next example.

For instance, let us consider the converse of Desargues' theorem (cf. [1]). Take ten points A, B, C, A', B', C', P, Q, R, O and nine lines[1] L_{ABP}, $L_{A'B'P}$, L_{ACQ}, $L_{A'C'Q}$, L_{BCR}, $L_{B'C'R}$, L_{PQR}, $L_{AA'O}$, $L_{BB'O}$. Impose the following incidence relations: for every line L_{XYZ}, it holds that $X \in L_{XYZ}$, $Y \in L_{XYZ}$, $Z \in L_{XYZ}$.

These twenty seven incidence conditions mean, in the algebraic case, that the triangles ABC and $A'B'C'$ are perspective with respect to line L_{PQR}, because this line contains the three intersection points of pairs of homologous sides of these triangles.

If Desargues converse statement holds –for instance, for points in $\mathbb{PK}^2$ (but it can be shown that it does not hold in general for points in $\mathbb{PT}^2$, see Example 2.5)– then these hypotheses imply that the triangles are, as well, perspective regarding point O, ie. that the three lines defined by pairs of homologous vertices meet at that point; or, to put this thesis in some more convenient way, that there exists a line $L_{CC'O}$ such that C, C', O belong to it.

Next, we can turn this theorem into a construction, in the following way. First, we view the whole theorem as a configuration of points and lines verifying certain incidence relations. Then, we intend to reproduce such configuration, starting, for instance, with six[2] arbitrarily given points A, B, C, A', B', X as input. Then proceed with the following steps:

[1] The reader is recalled that he/she should be careful about interpreting the following names of these lines: in principle they are just names without any intuitive meaning whatsoever, and these lines verify just the conditions imposed by the hypotheses below. In the tropical case, saying that there exists a line such that A, B, P belong to it, is not equivalent to saying that point P belongs to line AB, since we are always considering the *stable* line through A and B. It may happen that point P belongs to AB, but A does not belong to BP. So, to state in general terms (both for the classical and tropical cases) this converse of Desargues theorem, we have to be very careful.

[2] It seems that, with the above construction rules, this configuration can not be built up with a smaller number of points.

First, construct line AB, line AC, line BC, line AA', line BB' and line $A'B'$. Second, define the intersection points $O = AA' \cap BB'$ and $P = AB \cap A'B'$. Then, construct[3] line PX. Follow with the intersection points $Q = AC \cap PX$, and $R = BC \cap PX$. Construct the lines $B'R$, and $A'Q$ and, finally, take $C' = B'R \cap A'Q$.

By construction, it happens that PQR are on a line. So, triangles ABC and $A'B'C'$ are perspective with respect to line PQR. Thus, this construction "represents" the hypotheses of the converse of Desargues'. Hence, if Desargues converse statement holds (but it could be the case that this does not happen in the tropical case for some concrete input data), points C, C', O should be collinear.

2 Main result and examples

As stated in the introduction, given a tropical geometric construction with input elements $\{C_1, \ldots, C_n, p_1, \ldots, p_m\}$, we want to compute sufficient conditions for some given lifts of the curves C_i and points p_j, to be coherent with the construction, in the sense of verifying the same incidence relations.

Notice that every curve C_i has associated its Newton polygon Δ_i and hence, it may be written (Remark 1.1) by a polynomial $\widetilde{f_i} = \sum_{(k,l)\in\Delta_i\cap\mathbb{Z}^2} \widetilde{a}^i_{(k,l)} x^k y^l$. Also, every lift of an input point may be given by its coordinates $\widetilde{p}_j = (\widetilde{b}^j_1, \widetilde{b}^j_2)$. Take $N = \sum_{j=1}^n \#\{\Delta_j \cap \mathbb{Z}^2\} + 2m$. $\mathbb{K}^N$ can be considered as a configuration space (for the lifts of the input elements) which contains the coefficients of the defining polynomials of the lifted curves and the coordinates of the lifted points. Consider the map $\mathbb{K}^N \longrightarrow (\mathbb{Q}^a)^N$ such that, coordinate-wise, maps the element $\widetilde{a} = \alpha_r t^r + O(t^{r+\epsilon}) \mapsto \alpha_r$ into its principal coefficient.

In this situation the main theorem is the following:

Theorem 2.1. *Consider a concrete instance of a tropical plane geometric construction G with input elements $\{C_1, \ldots, C_n, p_1, \ldots, p_m\}$, curve C_i defined by a tropical polynomial $f_i =$ "$\sum_{(k,l)\in\Delta_j\cap\mathbb{Z}^2} a^i_{(k,l)} x^k y^l$ ", point $p_j = (b^j_1, b^j_2) \in (\mathbb{T}^*)^2$. Then we can compute a constructible set S in $(\mathbb{Q}^{a*})^N$ such that, if the vector $(Pc(\widetilde{a}^1_{(k,l)}), \ldots, Pc(\widetilde{a}^n_{(k,l)}), Pc(\widetilde{b}^1_1), \ldots, Pc(\widetilde{b}^m_2))$ of principal coefficients of a given lift of the input lies in S, then the same geometric construction G can be carried on the algebraic setting; moreover, its elements project onto the corresponding elements in the tropical construction.*

Notice we do not claim that there always exists such a well behaved lift for every construction. In fact, it is well possible that the computed S is empty, and, in this case, the theorem above does not give any information about the lifts. On the other hand, the theorem (as an existential statement) trivially holds, by taking S empty in every case. So the interest of the result lies in the concrete algorithm we develop to produce S, as we can show that it yields a sufficiently big S in many situations.

[3]This is a trick, in order to consider "an arbitrary line passing through P", which is not, strictly speaking, an allowable step, according to the very restrictive set of rules we have given above.

Example 2.2. Consider again the converse Desargues construction, with input elements $A = (13, 2)$, $B = (26, 21)$, $C = (-14, 3)$, $A' = (6, 9)$, $B' = (4, -16)$ $X = (1, -23)$. Then we find out (see next section for a summary description of the algorithm) that $S = (\mathbb{Q}^{a*})^{12}$. This means that for any collection of six points $\widetilde{A} = (\alpha_1^1 t^{-13} + O(t^{-13+\epsilon}), \alpha_2^1 t^{-2} + O(t^{-2+\epsilon}))$, $\widetilde{A'} = (\alpha_1^4 t^{-6} + O(t^{-6+\epsilon}), \alpha_2^4 t^{-9} + O(t^{-9+\epsilon}))$, etc. in $(\mathbb{K}^*)^{12}$, without any restriction on the α_i^j, we can repeat the construction scheme of converse Desargues for these new algebraic input points, obtaining the corresponding algebraic points $\widetilde{O}$, $\widetilde{P}$, $\widetilde{Q}$, $\widetilde{R}$ and $\widetilde{C'}$ and lines $\widetilde{A}\widetilde{B}$, $\widetilde{A}\widetilde{C}$, $\widetilde{B}\widetilde{C}$, $\widetilde{A}\widetilde{A'}$, $\widetilde{B}\widetilde{B'}$ $\widetilde{A'}\widetilde{B'}$, $\widetilde{P}\widetilde{X}$, $\widetilde{B'}\widetilde{R}$ and $\widetilde{A'}\widetilde{Q}$. Moreover, it will happen that for all the involved points or lines, say $\widetilde{Z}$, $T(\widetilde{Z}) = Z$.

We do not claim –as part of our main result– that it happens that the triangles $\widetilde{A}\widetilde{B}\widetilde{C}$ and $\widetilde{A'}\widetilde{B'}\widetilde{C'}$ are perspective with respect to $\widetilde{O}$, but in this concrete case it will be true, In fact, since the converse Desargues theorem holds in the algebraic case, it will hold for the lifted tropical elements, so they will form a configuration of triangles in perspective with respect to point $\widetilde{O}$. Then, since the projection of triangles in perspective with respect to a point yields triangles with the same property in the tropical setting, we conclude that for this particular instance, converse Desargues holds.

Example 2.3. For different input points, we may find that the obtained set S can be quite diverse. Eg. take now as input points $A = (1, 0)$, $B' = (-15, -4)$, $C = (-10, -8)$, $A' = (-18, -19)$, $B' = (14, -25)$, $X = (-15, 2)$. In this case we obtain that (with the same notation as above for the principal coefficients of the lift) $S = \{\alpha_1^1\alpha_2^4 - \alpha_2^1\alpha_1^4 \neq 0\}$ in $(\mathbb{K}^*)^{12}$. The theorem does not give information for the case that the principal coefficients do not lie in S. But a careful look into this concrete example shows that if the principal coefficients of the input are not in S, then the line $\widetilde{A'}\widetilde{Q}$ will not tropicalize onto $A'Q$, but onto another tropical line. So the computed set S is, in this case, maximal with respect to the property of lifting compatibility with the construction.

Example 2.4. As a totally negative example to the existence of a lift, consider $A = (7, 21)$, $B = (-24, -13)$, $C = (6, 3)$, $A' = (12, 8)$, $B' = (25, -4)$, $X = (-17, -23)$. In this case the computed S is empty. A more detailed analysis shows here that a lift never exists, since line $\widetilde{A'}\widetilde{Q}$ will never project onto the tropical line $A'Q$. So, again, our algorithm computes a maximal S in this case, too. However, the conclusion of converse Desargues' theorem (points C, C', O are collinear) holds for this input.

Example 2.5. Finally, consider the set of input points $A = (-24, -11)$, $B = (23, 24)$, $C = (-14, -2)$, $A' = (-7, 9)$, $B' = (25, 24)$, $X = (-5, 1)$. Now we find out again that $S = \emptyset$, but here the thesis of converse Desargues does not hold. That is, points C, C' and O do not lie on a tropical line. From this we deduce immediately that there cannot be a lift for the construction (and, thus, our computed set S is, again, maximal).

3 Summary description of the proposed method

We will summarily describe now the computation of the set $S \subseteq (\mathbb{Q}^a)^N$ involved in Theorem 2.1.

Assume it is given a tropical geometric construction G with input elements $\{C_1, \ldots, C_n, p_1, \ldots, p_m\}$, curve C_i with Newton polygon Δ_i, defined by

$$f_i = \text{“} \sum_{k=(k_1,k_2)\in\Delta_i\cap\mathbb{Z}^2} a_k^i x^{k_1} y^{k_2} \text{ ”}, \; p_j = (b_1^j, b_2^j).$$

Set $\widetilde{f_i} = \sum_{k=(k_1,k_2)\in\Delta_i\cap\mathbb{Z}^2} \alpha_k^i t^{-a_k^i} x^{k_1} y^{k_2}$, $\widetilde{p}_j = (\beta_1^j t^{-b_1^j}, \beta_2^j t^{-b_2^j})$, with $\{\alpha_k^i, \beta_k^j\}$ variables. This is a kind of generic lift, so that S will be defined in terms of $\{\alpha_k^i, \beta_k^j\}$. Let us start towards the computing S with the auxiliar constructible set Z in $(\mathbb{Q}^a)^N$ defined by $\{\alpha_k^i \neq 0, \beta_k^j \neq 0\}$. For each step of the construction G do the following:

If we are in the case of computing the curve $\widetilde{C}_i$ of Newton polygon Δ_i passing through $\widetilde{q}_1, \ldots, \widetilde{q}_j$, $j = \#(\Delta_i \cap \mathbb{Z}^2)$, this step is just solving a linear equation system. The unknowns of the system are the coordinates of a defining polynomial $\widetilde{f_i}$ and the linear restrictions are those induced by the condition of passing through the points. The coefficients of $\widetilde{f_i}$ are rational functions in the coefficients of the points. We provide, in [6], sufficient conditions for linear equation systems to be compatible with tropicalization. These conditions imposed to the principal coefficients are inequalities of the form $\widetilde{h}_i \neq 0$, $1 \leq i \leq M$, where the $\widetilde{h}_i$ are the pseudodeterminants associated to the linear system (cf. [6] for definitions and computations). These pseudodeterminants are polynomials in the principal coefficients of the coordinates of the points and basically ensure that the order of the principal coefficients of $\widetilde{f_i}$ does not increase. Append to the definition of Z the inequalities $\{\widetilde{h}_1 \neq 0, \ldots, \widetilde{h}_{j+1} \neq 0\}$ and set $\widetilde{C}_i$ as the solution to the linear system.

For the other construction step, the intersection of two curves $\widetilde{C}_{i_1}$, $\widetilde{C}_{i_2}$ with Newton polygon Δ_{i_1}, Δ_{i_2}, represented by $\widetilde{f}_{i_1}$, $\widetilde{f}_{i_2}$, set $M = \mathcal{M}(\Delta_{i_1}, \Delta_{i_2})$, the Minkowski sum of the polygons (this is the number of intersection points in $(\mathbb{K}^*)^2$ of two generic polynomials with Newton polygon Δ_{i_1}, Δ_{i_2}). Compute the two tropical resultants $H(y) = \mathrm{Res}_x(f_{i_1}, f_{i_2}) = \text{“}\sum_{r=0}^{R_1} h_r y^r \text{ ”}$, $P(x) = \mathrm{Res}_y(f_{i_1}, f_{i_2}) = \text{“}\sum_{r=0}^{R_2} p_r x^r \text{ ”}$, and the intersection set W of these resultants with the tropical curves. Compute a natural number a such that $x - ay$ is injective in W. Make now the change of variables $x = zy^a$ and compute the resultant $Q(x,y) = \mathrm{Res}_y(f_{i_1}(zy^a, y), f_{i_2}(zy^a, y)) = \text{“}\sum_{r=0}^{R_3} q_r z^r \text{ ”} = \text{“}\sum_{r=0}^{R_3} q_r x^r y^{-ar} \text{ ”}$. These three resultants determine uniquely the stable intersection of the given curves.

Compute $(b_1^1, b_1^2), \ldots, (b_M^1, b_M^2)$, the tropical stable intersection of C_{i_1}, C_{i_2} with multiplicities. Compute the algebraic resultants $\widetilde{H}(y) = \mathrm{Res}_x(\widetilde{f}_{i_1}, \widetilde{f}_{i_2}) = \sum_{r=0}^{R_1} \widetilde{h}_r y^r$, $\widetilde{P}(x) = \mathrm{Res}_y(\widetilde{f}_{i_1}, \widetilde{f}_{i_2}) = \sum_{r=0}^{R_2} \widetilde{p}_r x^r$ and, after the same change of coordinates $x = zy^a$, the resultant $\widetilde{Q}(x,y) = \mathrm{Res}_y(\widetilde{f}_{i_1}(zy^a, y), \widetilde{f}_{i_2}(zy^a, y)) = \sum_{r=0}^{R_3} \widetilde{q}_r z^r = \sum_{r=0}^{R_3} \widetilde{q}_r (xy^{-a})^r$. Write $\widetilde{h}_r = \overline{h}_r t^{-h_r} + O(t^{-h_r+\epsilon})$, $\widetilde{p}_r = \overline{p}_r t^{-p_r} + O(t^{-p_r+\epsilon})$, $\widetilde{q}_r = \overline{q}_r t^{-q_r} + O(t^{-q_r+\epsilon})$. $\overline{h}_r$, $\overline{p}_r$, $\overline{q}_r$ are polynomials in the principal coefficients of $\widetilde{f}_{i_1}$, $\widetilde{f}_{i_2}$. Take $2M$ new variables $\{\gamma_1^1, \gamma_2^1, \ldots, \gamma_M^1, \gamma_M^2\}$. Add to the definition of Z the restrictions $\gamma_i^j \neq 0$, $1 \leq i \leq M$, $1 \leq j \leq 2$ and the restrictions $\overline{h}_r \neq 0$, $\overline{p}_r \neq 0$, $\overline{q}_r \neq 0$, the principal coefficients of the coordinates of the computed resultants. With these restrictions, the algebraic resultants $\widetilde{H}$, $\widetilde{P}$, $\widetilde{Q}$ will tropicalize onto the tropical resultants H, P, Q and the intersection points of the lifts that verify this restrictions will tropicalize onto the corresponding stable intersection. For every point (b_i^1, b_i^2)

write $\widetilde{H}(yt^{-b_2^i}) = \overline{H}_i(y)t^{-H(b_2^i)} + O(t^{-H(b_2^i)+\epsilon})$, $\widetilde{P}(xt^{-b_1^i}) = \overline{P}_i(x)t^{-P(b_1^i)} + O(t^{-P(b_1^i)+\epsilon})$ and $\widetilde{Q}(xt^{-b_1^i}, yt^{-b_2^i}) = \overline{Q}_i(x,y)t^{-Q(b_1^i,b_2^i)} + O(t^{-Q(b_1^i,b_2^i)+\epsilon})$. Add to Z the restrictions needed to assure that the γ_j^i are the principal coefficients of the intersection points of the curves. Namely, add $\overline{H}_i(\gamma_2^i) = 0$, $\overline{P}_i(\gamma_1^i) = 0$ and $\overline{Q}_i(\gamma_1^i, \gamma_2^i) = 0$, $i \leq i \leq M$. Add also the following condition in order to avoid the case of multiple roots, $\bigwedge_{i \neq j}(\gamma_i^1 \neq \gamma_j^1 \vee \gamma_i^2 \neq \gamma_j^2)$. Set $\widetilde{q}_1 = (\gamma_1^1 t^{-b_1}, \gamma_1^2 t^{-b_2}), \ldots, q_M = (\gamma_M^1 t^{-b_1}, \gamma_M^2 t^{-b_2})$ and proceed with the next step. This points $\widetilde{q}_i$ do not need to be the intersection points of the algebraic curves, but they will have the same principal coefficient.

We have built up a constructible set Z in $(\mathbb{Q}^a)^L$. This corresponds with principal coefficient points of some possible lifts of the whole construction. Finally, our goal, the set S is defined as the projection of Z over the variables $\{\alpha_k^i, \beta_k^j\}$. For more details concerning this method and its correctness we refer to [7].

References

[1] Berger, M. Geometry, Vol. 1 and 2. Springer, 1987

[2] Einsiedler, M. Kapranov, M. Lind, D. Non-archimedean amoebas and tropical varieties. Preprint AIM 24 -31, http://arxiv.org/abs/math/0408311, 2000.

[3] Itenberg, I. Kharlamov, V. Shustin, E. Welschinger invariant and enumeration of real rational curves, Int. Math. Res. Not. 2003, no. 49, 2639-2653, 2003

[4] Mikhalkin, G. Enumerative tropical algebraic geometry in $\mathbb{R}^2$, J. Amer. Math. Soc. no.2, 313-377, 2005.

[5] Richter-Gebert, J. Sturmfels, B. Theobald, T. First steps in tropical geometry. Idempotent mathematics and mathematical physics, 289-317, Contemp. Math., 377, Amer. Math. Soc., Providence, RI, 2005

[6] Tabera, L. F. Tropical Pappus' theorem, Int. Math. Res. Not. 2005 no. 39 2373-2389, 2005.

[7] Tabera, L. F. Tropical plane geometric constructions, submitted 2005, http://arxiv.org/abs/math/0511713

Luis Felipe Tabera
Departamento de Matemáticas, Estadística y Computación
Universidad de Cantabria, Spain
luisfelipe.tabera@unican.es
http://personales.unican.es/taberalf

The author is supported by the European Research Training Network RAAG (HPRN-CT-2001-00271), a FPU research grant and the project MTM2005-08690-C02-02 from the Spanish Ministerio de Educación y Ciencia.

A Bus-Based Semi-Completely-Connected Network for High-Performance On-Chip Systems

Masaru Takesue

Abstract

This paper proposes a network, called *BSK-cube*, to alleviate the long-wire and pin-neck problems against high-performance on-chip systems through a *small diameter* and *dynamic clustering*. We derive a 2^n-node recursive *semi-complete graph*, $\mathrm{SK}_n^\gamma(v_1, \ldots, v_\gamma, w_\gamma)$ $(\sum_{i=1}^{\gamma} v_i + w_\gamma = n)$, from the relationship between the Hamming code-based partitions [3]. The BSK_n^γ-cube has the SK_n^γ topology whose constituent SK's links incident to a node are replaced by a single bus for the node; so each node is connected to γ levels of buses. The diameter of BSK^γ-cube equals γ. The bus length is $\mathbf{O}(\sqrt{2^{n_\gamma}})$ $(n_\gamma = v_\gamma + w_\gamma)$ for the level-γ (i.e., the local) bus, and it is $\mathbf{O}(\sqrt{2^n})$ for the level-1 (i.e., the global) bus. We show by analysis that a few basic operations such as simultaneous distance-d exchange $(1 \leq d \leq n)$ are performed in γ bus-steps without bus congestion. The dynamic clustering of memory requests reduces the delay of off-chip memory requests, as compared with the static clusters fixed in hardware.

Key words: On-chip networks, Hamming codes, partitions, semi-complete graphs.

Introduction

Future LSI technologies will allow billions of transistors to be located on a single chip, so that a large portion of an on-chip multiprocessor (CMP) will be accommodated in the chip. However, the signal delay due to long wires will dominate the clock cycle time of the CMP if the feature size of wires will scale with the same pace as for the transistors [1]. Moreover, the CMP design may be restricted by the number of I/O pins on the chip periphery. The number of pins is currently of the order of 1k, e.g., equal to about 1400 for the UltraSPARC III chip [2], and this number will not increase as the same rate as the transistor size will decrease. Then with the limited number of pins, we have to maintain a high traffic rate required between the on-chip caches and (probable) off-chip memory.

To cope with the long-wire and pin-neck problems against high-performance on-chip systems, we have proposed three networks [3]-[5]. The scheme underlying those networks is the *sets of partitions* produced with *multiple suits of codewords* of the extended Hamming code [3]. The ideas behind our approach are to alleviate the long-wire problem with a *small diameter*, and the pin-neck problem through *dynamic clustering*. We expect a small diameter to be effectively equivalent to short wires.

In the dynamic clustering, a set of clusters is produced for each off-chip target such as a memory block when the requests are sent to the *leaders* (i.e., the representative nodes) of

the partitions to which the requesting nodes belong; each leader then sends a single request for the sake of the received requests to the target. So the traffic on the chip boundary reduces in the same way as with the static clusters fixed in hardware. Moreover, the sets of clusters, and hence, the leaders for a specific node are generally different for separate targets, so that the traffic to a leader reduces as compared with in the static cluster.

The psi-cube [5] is a bus-based extended hypercube. A problem of the 2^n-node psi-cube is its rather large diameter, $\lfloor n/2 \rfloor$, as a bus-based network. Another problem is that given an n, the suit and partition sizes, 2^k and 2^p, are fixed, so the network organization and its layout are restricted, where p and k $(= n - p)$ are the parity and information sizes of the n-bit address; the other Hamming code-based networks [3]-[4] have the same problems.

To further mitigate the long-wire problem with a much smaller diameter, this paper proposes a network, called *BSK-cube*. This network has the topology of semi-complete (SK) graph of which point-to-point links incident to a node are replaced by a single bus owned by the node. We derive the SK graph from the relationship among the suits of codewords and the sets of partitions produced with the all suits. The 2^n-node SK graph has 2^p suits each of 2^k nodes. The nodes in each suit are configured into the complete (K) graph, every node of which is connected to the $2^p - 1$ nodes in the other K graphs.

To remove the restrictions of the Hamming code-based networks mentioned above, we modify the SK graph so that we can choose any values for k and p as long as $k + p = n$. Moreover, to reduce the length, especially of the local buses as mapped on the BSK-cube, we recursively convert the K components of the SK into smaller-sized SK graphs, leading to a recursive SK graph, $\mathrm{SK}^\gamma(v_1, \ldots, v_\gamma, w_\gamma)$, where $\gamma \geq 1$ and $\sum_{i=1}^{\gamma} v_i + w_\gamma = n$.

The diameter of the BSK^γ-cube is equal to γ (note: $\gamma = 1$ for the non-recursive BSK-cube), and the length of local buses equals $2^{v_\gamma} + 2^{w_\gamma}$. Although the length of global buses is $\mathbf{O}(2^{n/2})$, we can use a larger feature size for global buses than for local ones [14] since the layout patterns for the buses are very regular. Then the signal delay along a long bus will be not so large as reported in [1].

To show the BSK^γ-cube's potential, we analyze a few basic communication operations such as simultaneous distance-d exchange $(1 \leq d \leq n)$ and data streaming, and show that those are performed in γ bus-steps without bus contention. With the dynamic clustering as applied to a memory hierarchy in a CMP, the traffic to leaders of clusters reduces by a factor of at most 2^p, compared with the traffic in the static clusters.

It should be noted that based on the original SK graph, we can also obtain a BSK-cube and its recursive version: Then its diameter and performance are the same as those of the networks based on the modified SK graph, though the aspect ratio (i.e., the ratio of the width to the height) of the layout is restricted and the layout complexity increases.

Related work: Commercial multiprocessors, STiNG [6] and SGI Origin [7], adopt the clusters connected by the ring and the fat bristled hypercube, respectively. A research CMP, Hydra [8], has 4 processors and exploits two buses to connect their level-1 and level-2 caches with each other, while another CMP, Piranha [9], uses a crossbar between the level-1 and level-2 caches for 8 processors. A reconfigurable chip includes 64 simple processing nodes interconnected by the mesh [10]. No dynamic clusters are found in those systems.

For the networks on a chip (NoCs), tree-type [11] and mesh-type [12] networks are

generally desirable because of their short wires and easy layout. Another such topology is a recursive ring network where a recursive level is an extended ring of 8 nodes with extra links between the nodes located at the opposite locations on the ring [13]. A hierarchical bus network [14] is also attractive since it reduces local communication overhead but also allows us to use a standard cache coherence protocol implemented with buses. Our BSK$^\gamma$-cube is a recursive bus network where each node is connected to multiple buses.

From the viewpoint of network modeling, the group-theoretical model [15] organizes a network so that given a set of generator for a finite group G, there is an edge from an element a to an element b if and only if there is a generator g such that $ag = b$ in the group G. The model can represent most symmetric networks. The n-dimensional linear recursive networks [16] are associated with a linear recurrence of the form $X_n = a_1X_{n-1}+a_2X_{n-2}+\cdots+a_kX_{n-k}$, that is interpreted such that an n-dimensional network X_n is organized with a_i number of $(n-i)$-dimensional subnetworks X_{n-i} $(i = 1, 2, \ldots, k)$. Our model for organizing the BSK-cube and the previous networks [3]-[5] is primarily for designing novel networks based on the partitions and can represent the mesh [4], hypercube [5], and so on.

In the rest of paper, Section 1 introduces the SK$^\gamma$, after summarizing the Hamming code-based partitioning. Section 2 presents the structure, layout, and routing of the BSK$^\gamma$-cube. Section 3 discusses on the properties of the network preferable for parallel computation. Section 4 summarizes the paper and discusses future research. Appendix A shows a BSK-cube obtained with the original SK, for reference.

1 Semi-Complete Graphs

This section summarizes the Hamming code-based partitioning, and defines the SK graph, modified SK graph, and its recursive version, SK$^\gamma$.

1.1 The partitioning

A *codeword* w of the n-bit *Hamming code* $\psi(n, k)$ has p-bit parity $(p = n - k)$ for k-bit information, where p is the smallest integer satisfying $(2^p - 1) \geq n$. The $p \times n$ parity-check matrix H_n is required for encoding codewords and decoding received words. When received a word w', we calculate the p-bit *syndrome* $\varepsilon = w' \cdot H_n^t$ (where H_n^t is the transpose of H_n) and decode it into the n-bit *error vector* e_ε. The syndrome ε indicates the erroneous bit-position if the word w' has a single-bit error, so its error vector e_ε has then a single bit of 1 in the erroneous position. The codeword w for the received word w' with a single-bit or no error is obtained by $w' \oplus e_\varepsilon$, where $\oplus$ is the bitwise exclusive-OR operation.

We assume that all errors are single-bit or double-bit ones, and exploit both of them in the partitioning. Generally, a double-bit error cannot be corrected since the pair (d, f) $(f > d)$ of erroneous bit positions is not unique for an $\varepsilon = d \oplus f$. As far as used in the partitioning, this problem can be resolved by fixing position f equal to 2^{p-1}; then position d is obtained by $\varepsilon \oplus f$. Let P_w denote the partition represented by the codeword w.

In the partitioning, we put a word w' whose syndrome equals ε into the partition P_w such that $w = w' \oplus e_\varepsilon$. Then 2^k *partitions each of* 2^p *words* are produced. More importantly,

we produce another set of partitions, called *suits* $\mathcal{S} = \{S_0, S_1, \ldots, S_{2^p-1}\}$ *of codewords*, by $S_0 \oplus e_\varepsilon$ for all error vectors $e_\varepsilon \in E$, where $E = \{0, 1, \ldots, 2^p - 1\}$ is the set of all error vectors. In this case, we obtain 2^p *suits each of* 2^k *codewords*.

In the rest of the paper, we discuss on the node space, so we use term *nodes* instead of words and *leaders* in place of codewords. For instance, we restate so that partition P_ℓ represented by leader ℓ consists of nodes $\ell \oplus E$, $\mathcal{S}$ is the suits of leaders, and so forth.

In this paper, we use the following interesting relationships between the suits of leaders and the partitions produced with the *all suits*: The first relationship is used to obtain a *semi-complete (SK) graph*, and the second one in the routing for the dynamic clustering.

1) *The* 2^p *members of one partition are located in the different* (2^p) *suits.*

2) *Leader* ℓ *of the partition* P_ℓ *to which a node* s *belongs when partitioned with the suit* $S_{\ni t}$ *that includes a node* t *is obtained by* $\ell = s \oplus e_\varepsilon$*, where* ε *is the syndrome for* $s \oplus t$*.*

For example, the first relationship in the 3-bit address space is shown in Fig. 1. The lines incident to a node mean that it is the leader of the partition of which members (excluding the leader) are at the other ends of the lines. Since $p = 2$ and $k = 1$ in this case, there are 4 $(= 2^p)$ sets of partitions produced respectively with 4 suits, S_i $(i = 0, 1, 2, 3)$, of leaders. One set has 2 $(= 2^k)$ partitions each of 4 $(= 2^p)$ members. Notice that the members of each partition, for instance $P_{000} = \{000, 100, 010, 001\}$, are distributed in the all suits.

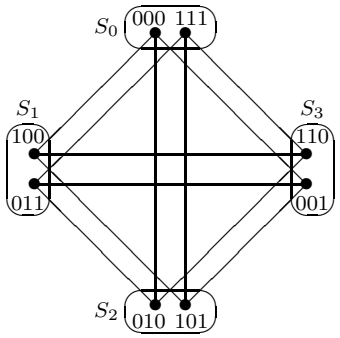


Fig. 1. The suits and partitions in the 3-bit address space.

1.2 SK graphs

Although the completely-connected network (i.e., the K-graph) has the diameter equal to 1, it needs the most complex layout due to the greatest number of links of the direct-connect networks such as the hypercube and mesh; the 2^n-node K-graph, K_n, has $2^n - 1$ links per node. To reduce the number of links, preserving the diameter as small as possible, we introduce an *SK graph*. The 2^n-node SK graphs, SK_n, has the same topology as the one represented by the first relationship described in Section 1.1, except that each suit of leaders is configured into a complete graph K_k as defined below.

Definition 1.1. An SK_n graph has 2^p number of K_k-graphs, and each node in every K_k has one link to a node in each of the other K_k-graphs.

The SK_n has $(2^p + 2^k - 2)$ links per node, and hence, a total of $2^n(2^p + 2^k - 2)/2$ links, leading us to an easier layout as compared with the K_n that has $2^n(2^n - 1)/2$ links. For example, Fig. 2 shows the SK_3 obtained from the relationship shown in Fig. 1 by connecting the nodes in every suit with each other to produce a K_1.

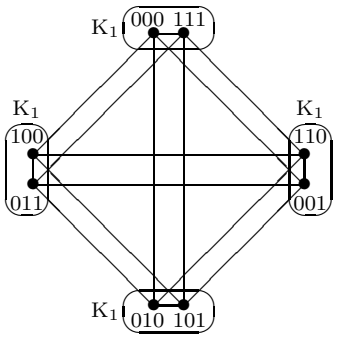


Fig. 2. The SK_3 derived from the relationship shown in Fig. 1.

We will present the layout of a bus-based SK network, i.e., the BSK-cube, in which the nodes are arranged into an array in Section 2. Concerning the layout, a remarkable weak point of the SK described above is the difficulty in putting the members of each partition in one row (or one column), leading to longer buses. For shorter buses, we introduce *simple partitions* as defined below and remap the node addresses onto the SK topology so that the members are arranged in one row or column of the array. Let v and w be the upper v bits and the lower w ($w = n - v$) bits of the n-bit node address, and s and ℓ be their values. We denote the node address by $\langle s, \ell \rangle$. Then the simple partitions are defined as follows.

Definition 1.2. The suit S_s ($0 \le s \le 2^v - 1$) consists of the 2^w leaders $\langle s, * \rangle$ ($* = 0, 1, \ldots, 2^w - 1$), and the partition $P_{\langle s,\ell \rangle}$ has the members $\langle *, \ell \rangle$ ($* = 0, 1, \ldots, 2^v - 1$).

It is clear from Definition 1.2 that the set of suits S_s ($s = 0, 1, \ldots, 2^v - 1$), as well as the set of partitions $P_{\langle s,* \rangle}$ represented by the leaders $\langle s, * \rangle$ in one suit S_s, organizes the partitions of the n-bit address space, and that the members $\langle *, \ell \rangle$ of one partition $P_{\langle s,\ell \rangle}$ are distributed to the all suits. Thus the simple partitions preserve these properties from the Hamming code-based partitions, and hence, we can produce an SK graph based on the simple partitions. Let $SK_n(v, w)$ ($v + w = n$) denote the SK_n consisting of 2^v suits each of 2^w leaders; then the size of a partition is equal to 2^v.

For instance, the $SK_3(2, 1)$ with the simple partitions is shown in Fig. 3. The node address $\langle a_0 a_1, a_2 \rangle$ is denoted by $a_0 a_1 a_2$, for space. The nodes in each suit S_s ($0 \le s \le 2^v - 1 = 3$) are each organized into a K_1 graph. The suits are $S_{00} = \{\langle 00, 0 \rangle, \langle 00, 1 \rangle\}$, $S_{01} = \{\langle 01, 0 \rangle, \langle 01, 1 \rangle\}$, $S_{10} = \{\langle 10, 0 \rangle, \langle 10, 1 \rangle\}$, and $S_{11} = \{\langle 11, 0 \rangle, \langle 11, 1 \rangle\}$. The partitions

$P_{\langle 01,*\rangle} = \{P_{\langle 01,0\rangle}, P_{\langle 01,1\rangle}\}$ represented by the leaders $\langle 01,*\rangle$ in, for example, suit S_{01} organize a set of partitions in the 3-bit space. Notice that the members $\langle 11,0\rangle$, $\langle 00,0\rangle$, $\langle 01,0\rangle$, and $\langle 10,0\rangle$ of, for example, partition $P_{\langle 11,0\rangle}$ are scattered to the all suits.

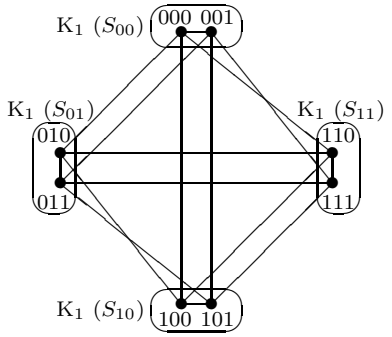


Fig. 3. The $\mathbf{SK}_3(2,1)$ based on the simple partitions.

We next present a remarkable property of the SK with the simple partitions that contributes to a performance improvement as well as an easy layout, as compared with the SK with the Hamming code-based partitions.

Theorem 1.3. *In the $SK_n(v,w)$ graph, the members of each partition organize a K_v.*

Proof. Let $M = \{\langle 0,\ell\rangle, \langle 1,\ell\rangle, \cdots, \langle 2^v-1,\ell\rangle\}$ be the members of a partition $P_{\langle s,\ell\rangle}$. Then the partition $P_{\langle s',\ell\rangle}$ represented by a member $\langle s',\ell\rangle \in M$ has the same members M. Thus the members M of $P_{\langle s,\ell\rangle}$ are connected to each other, and hence, organize a K_v. □

For example (see Fig. 3), partitions $P_{\langle 11,0\rangle}$, $P_{\langle 00,0\rangle}$, $P_{\langle 01,0\rangle}$, and $P_{\langle 10,0\rangle}$ that are represented respectively by the members $M = \{\langle 11,0\rangle, \langle 00,0\rangle, \langle 01,0\rangle, \langle 10,0\rangle\}$ of partition $P_{\langle 00,0\rangle}$ have the same members M, that configure a K_2.

Theorem 1.3 does not hold for the original SK derived from the Hamming code-based partitions. It has the same number of links but increases its layout complexity as compared with the SK with the simple partitions. Owing to Theorem 1.3, we can arrange the leaders $\langle s,*\rangle$ in suit S_s on the row s of a $2^v \times 2^w$ array, and the members $\langle *,\ell\rangle$ of partition $P_{\langle s,\ell\rangle}$ on the column ℓ of the array as presented in Section 2.

Moreover, we can choose the v and w values for an n, while given an n, the sizes p and k are fixed in the Hamming code-based partitions. So the aspect ratio of the layout for $\mathrm{SK}_n(v,w)$ can be adjustable. In the rest of the paper, the SK stands for the one based on the simple partitions if not mentioned otherwise.

1.3 Recursive SK graphs

For a large n, the size 2^w of the K_w component of an $\mathrm{SK}_n(v,w)$ may be too large to achieve an easy layout. To reduce the layout complexity, we convert the K_w into an SK: Let $n_1 = n$, $v_1 = v$, $w_1 = w$, $v_i + w_i = n_i = w_{i-1}$, and $\sum_{i=1}^{\gamma} v_i + w_\gamma = n$. Then we can generally produce a γ-level recursive SK_n, denoted by $\mathrm{SK}_n^{\gamma}(v_1,\ldots,v_\gamma,w_\gamma)$, in the following way.

Definition 1.4. The $\mathrm{SK}_n^\gamma(v_1, \ldots, v_\gamma, w_\gamma)$ is produced from $\mathrm{SK}_n^1(v_1, w_1) = \mathrm{SK}_n(v, w)$ by repeating the next procedure for $i = 2, \ldots, \gamma$: Convert each $\mathrm{K}_{w_{i-1}}$ component of the $\mathrm{SK}_n^{i-1}(v_1, \ldots, v_{i-1}, w_{i-1})$ into the $\mathrm{SK}_{n_i}(v_i, w_i)$ graph to obtain the $\mathrm{SK}_n^i(v_1, \ldots, v_i, w_i)$.

For instance, the $\mathrm{SK}_5^2(2, 2, 1)$ (see Fig. 4) is produced by converting each of the K_3 components of $\mathrm{SK}_5^1(2, 3)$ into the $\mathrm{SK}_3(2, 1)$ shown in the large oval, where $\gamma = 2$, $v_1 = v_2 = 2$, and $w_\gamma = w_2 = 1$, so that $\sum_{i=1}^{2} v_i + w_2 = 5 = n$.

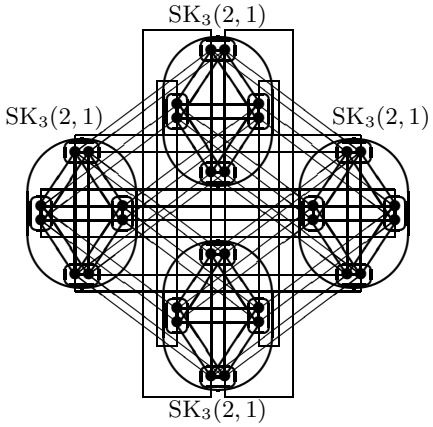


Fig. 4. The $\mathrm{SK}_5^2(2, 2, 1)$ of which components are $\mathrm{SK}_3(2, 1)$'s.

The SK_n^γ has $\sum_{i=1}^{\gamma}(2^{v_i} - 1) + (2^{w_\gamma} - 1)$ links per node, so when $v_i = v_j = w_\gamma$ $(i \neq j)$, the total number of links equals $(\gamma+1)(2^{(\gamma+2)n/(\gamma+1)} - 2^{n-1})$, that is $\mathbf{O}(2^n)$ when γ is large. As compared with the complete graph K_n, the SK_n^γ reduces the number of links by a factor of about 2^n with the increased diameter of $\gamma + 1$.

2 The BSK-Cube

This section first describes the structure, layout, and routing of the bus-based SK and SK^γ networks, respectively called *BSK-cube* and *BSK$^\gamma$-cube*, and next analyzes the performance of a few basic communication operations preferable for efficient parallel computations to show the potential of the networks. The effect of dynamic clustering as applied to a memory hierarchy is also analyzed.

2.1 Structure of the BSK-cube

In organizing the BSK_n-cube from the SK_n in the way defined below, we use bus connections for the BSK-cube instead of the SK's point-to-point links to reduce the number of routing steps required in the BSK-cube to one half of the SK's, as well as the number of wires to $\mathbf{O}(2^{3n/2})$, with the penalty of a decreased bandwidth and an increased latency. So the BSK_n-cube has the diameter equal to 1 bus-step. The BSK_n-cube is organized as follows:

Definition 2.1. The $\mathrm{BSK}_n(v, w)$-cube is obtained if we replace all links that are incident to a node by a single bus for the node, for all nodes in the $\mathrm{SK}_n(v, w)$ graph.

From Definition 2.1, the bus for node ℓ connects not only the members of partition P_ℓ but also the nodes in the $\mathrm{K}_w \ni \ell$ to each other. We lay out the BSK_n-cube as defined below, for as small a maximum bus length and as regular a layout as possible.

Definition 2.2. We configure the nodes of the $\mathrm{BSK}_n(v, w)$-cube into a $2^v \times 2^w$ array and put node $\langle s, \ell \rangle$ at the position of array index (s, ℓ).

The layout of $\mathrm{BSK}_6(3, 3)$-cube is shown in Fig. 5; for reference, the layout of BSK_6-cube based on the original SK graph derived from the Hamming code-based partitions is shown in Fig. A1 in Appendix A. The $\mathrm{BSK}_6(3, 3)$-cube has 64 buses, one per node. The two-digit integer $s\ell$ in the box stands for the node address $\langle s, \ell \rangle$.

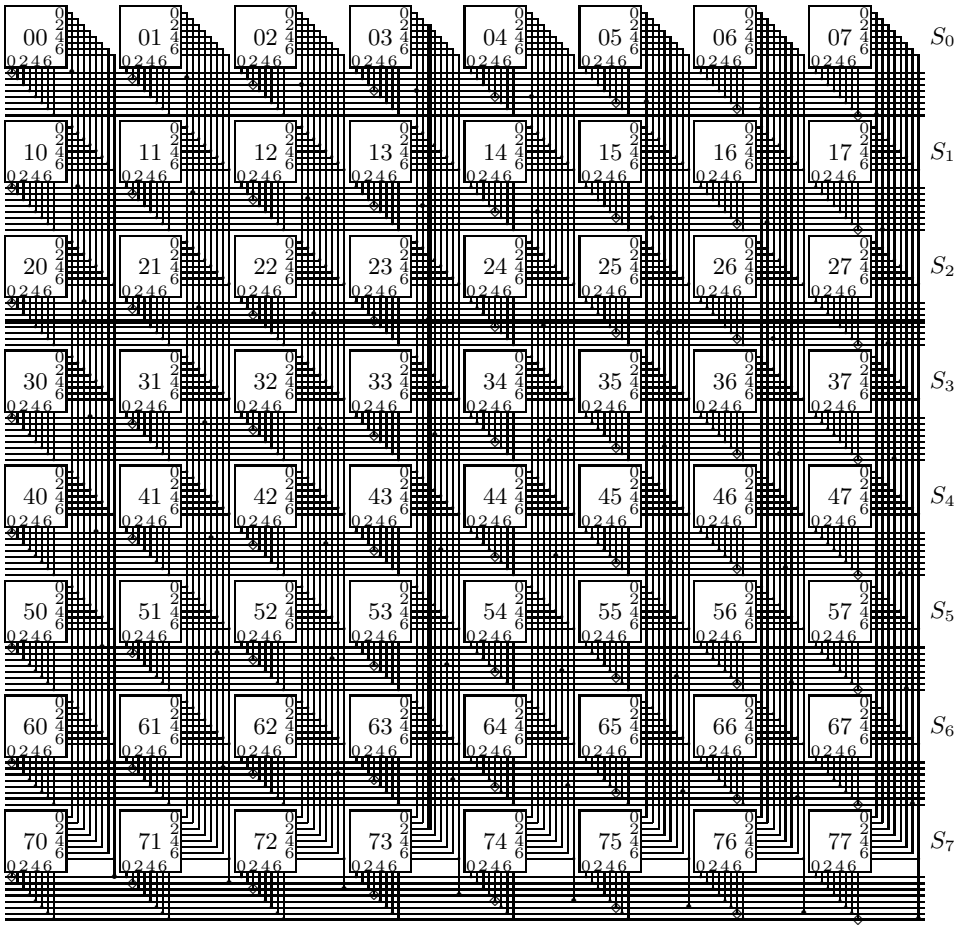


Fig. 5. The layout of the $\mathrm{BSK}_6(3, 3)$-cube.

Each node has two sets of ports; one set for vertical lines, and the other for horizontal lines; for space, only even-numbered ports are shown. The bus for node $\langle s, \ell \rangle$ consists of a horizontal line along row s and a vertical line along column ℓ; for convenience, the diamond at the cross point of the horizontal line and the stub to node $\langle s, \ell \rangle$ shows that the node is the owner of the bus. The bus of node, for instance, $\langle 2, 3 \rangle$ runs along row 2 and column 3 as shown by the bold line. The K_3 for suit S_s are organized by the 8 buses along row s, while the K_3 for partitions $P_{\langle *, \ell \rangle}$ is configured by the 8 buses along column ℓ.

2.2 Routing on the BSK$_n$-cube

Let S-ports (for choosing a suit) and L-ports (for selecting a leader in a suit) denote the ports connected to the vertical and horizontal lines of the buses, respectively, and $\langle s, \ell \rangle$ and $\langle s_t, \ell_t \rangle$ be the source and target addresses in a routing. Then the routing on the BSK-cube is performed in the following way, and its diameter is given by the next theorem.

Definition 2.3. To send a packet to the target node $\langle s_t, \ell_t \rangle$, the source node $\langle s, \ell \rangle$ puts the packet on its S-port s_t if $s \neq s_t$, or on its L-port ℓ_t otherwise.

Theorem 2.4. *The diameter of the BSK$_n$-cube is equal to one, independent of n.*

Proof. With the routing method of Definition 2.3, the packet is sent from the source to the target in 1 bus-step, because the S-port s_t of source $\langle s, \ell \rangle$ is connected to the bus of node $\langle s_t, \ell \rangle$ and that bus is connected with the target $\langle s_t, \ell_t \rangle$ if $s \neq s_t$, or because the L-port ℓ_t of source $\langle s, \ell \rangle$ is connected to the bus of the target $\langle s_t, \ell_t \rangle$ otherwise. □

For example, suppose that the source and target are nodes $\langle 2, 3 \rangle$ and $\langle 7, 6 \rangle$ in Fig. 5. Then the source node $\langle 2, 3 \rangle$ outputs the packet to S-port 7. The port is connected to the bus of node $\langle 7, 3 \rangle$, and the bus is connected with the target $\langle 7, 6 \rangle$. When the target is $\langle 2, 7 \rangle$ for the same source, the packet is sent via the source's L-port 7 that is connected to the target. In both cases, the packet is routed to the target in 1 bus-step.

Let the unit length of the bus equal the distance between the adjacent nodes on the BSK$_n$-cube layout, and the *bus length* be defined as the maximum of the distances for a signal to traverse for all buses. Then the distance is largest when the source and target are located at, for instance, the top-left and bottom-right corners of the array. Thus,

Theorem 2.5. *The bus length of the BSK$_n(v, w)$-cube equals $2^v + 2^w$, that reduces to $2 \cdot 2^{n/2}$ when $v = w$.*

2.3 Recursive BSK$_n$-cube

We organize a γ-level recursive BSK-cube, BSK$_n^\gamma$, from the SK$_n^\gamma(v_1, \ldots, v_\gamma, w_\gamma)$ so that it has short and long buses for local and global communications, respectively. The BSK$^\gamma$ has γ buses per node as described below:

Definition 2.6. We first produce one BSK$_{n_\gamma}(v_\gamma, w_\gamma)$ from each SK$_{n_\gamma}(v_\gamma, w_\gamma)$ as described in Definition 2.1, and lay out it into a $2^{v_\gamma} \times 2^{w_\gamma}$ array applying Definition 2.2. Next, for i

from $\gamma - 1$ to 1, we obtain one $\mathrm{BSK}_{n_i}^{\gamma-i+1}(v_i, \ldots, v_\gamma, w_\gamma)$ from each $\mathrm{SK}_{n_i}^{\gamma-i+1}(v_i, \ldots, v_\gamma, w_\gamma)$ by replacing its links remaining incident to a node with a single bus and lay out the 2^{v_i} number of $\mathrm{BSK}_{n_{i+1}}^{\gamma-i}$ cubes organizing one $\mathrm{BSK}_{n_i}^{\gamma-i+1}$ in one row or column.

For example, a skeletal layout of $\mathrm{BSK}_{10}^{3}(2,2,3,3)$-cube produced from the $\mathrm{SK}_{10}^{3}(2,2,3,3)$ is shown in Fig. 6. Each box stands for the $\mathrm{BSK}_6(3,3)$ that is the same as the one shown in Fig. 5, and the BSK_6's in the oval organize a $\mathrm{BSK}_8^2(2,3,3)$. The line segments incident to the left and at the top of a node (shown by the •) symbolize the level-3 (local) bus, i.e., the S-ports and L-ports of the BSK_6. The horizontal line is the level-2 (intermediate) bus to which the $2^{v_2} = 4$ nodes from the member-BSK_6's of a BSK_8 are connected, while the vertical line is the level-1 (global) bus for connecting the $2^{v_1} = 4$ nodes in the constituent BSK_8's to each other. So each node has three buses for levels 1 to 3; for space, only the level-1 buses for nodes $\langle 0, \ell \rangle$ ($\ell = 0, 64, 128, 192$) and the level-2 buses for nodes $\langle s, 0 \rangle$ ($s = 0, 1, 2, 3$) are depicted; note that the address notation for the BSK_{10} is used.

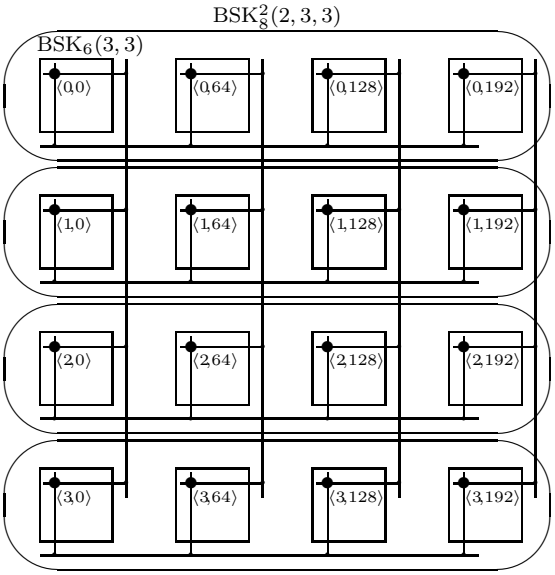


Fig. 6. A skeletal layout of the $\mathrm{BSK}_{10}^{3}(2,2,3,3)$-cube.

Let nodes $\langle s_1, \ldots, s_\gamma, \ell_\gamma \rangle$ and $\langle s_{t,1}, \ldots, s_{t,\gamma}, \ell_{t,\gamma} \rangle$ be the source and target in a routing on the $\mathrm{BSK}_n^\gamma(v_1, \ldots, v_\gamma, w_\gamma)$. Then we send the packet from node $\langle s_{t,1}, \ldots, s_{t,i-1}, s_i, \ldots, s_\gamma, \ell_\gamma \rangle$ to node $\langle s_{t,1}, \ldots, s_{t,i-1}, s_{t,i}, \ldots, s_\gamma, \ell_\gamma \rangle$ via the bus of the receiving node in the routing step i ($1 \le i < \gamma$) if $s_{t,i} \neq s_i$. In the last step (i.e., step γ), the packet is sent to the target node according to the routing method described in Definition 2.3. The diameter equals γ since the packet reaches the target at most in γ bus-steps.

For example, on the $\mathrm{BSK}_{10}^{3}(2,2,3,3)$-cube (see Fig. 6), suppose the routing from node $\langle 0,0,0,0 \rangle$ ($= \langle 0,0 \rangle$ in Fig. 6) in the top-left box to node $\langle 3,3,7,6 \rangle$ ($= \langle 3, 254 \rangle$) in the bottom-right box. Then the packet is sent along the path of nodes $\langle 0,0,0,0 \rangle$, $\langle 3,0,0,0 \rangle$ (=

$\langle 3,0\rangle$), $\langle 3,3,0,0\rangle$ ($=\langle 3,192\rangle$), and $\langle 3,3,7,6\rangle$), via the buses of nodes $\langle 3,0,0,0\rangle$, $\langle 3,3,0,0\rangle$, and $\langle 3,3,7,0\rangle$ ($=\langle 7,0\rangle$; this node in not shown in Fig. 6), respectively, in 3 bus-steps.

Last, let's analyze the bus length of $\mathrm{BSK}_n^\gamma(v_1,\ldots,v_\gamma,w_\gamma)$: Assume that γ is odd, $v_{2j-1}=v_{2j}$ $(1\le j\le(\gamma-1)/2)$, and $v_\gamma=w_\gamma$. Then the bus length is equal to $2^{v_\gamma}+2^{w_\gamma}$ in the level γ. Otherwise, the buses in the levels $(2j-1)$ and $2j$ have the same length $\Pi_{k=0}^{(\gamma+1)/2-j}2^{v_{\gamma-2k}}$. So the level-1 and level-2 (i.e., the global) buses have the length equal to $2^{v_\gamma}2^{v_{\gamma-2}}\cdots 2^{v_3}2^{v_1}=2^{n/2}$. For example, the BSK_{10}^3 layout (see Fig. 6) satisfies the assumption mentioned above, so that the bus length of the level-3 (local) bus equals $2^3+2^3=16$ units, and the length of the global buses (in the levels 1 and 2) are equal to $2^{10/2}=32$ units.

3 Properties of the BSK-cube

This section analyzes the exchange, streaming, and dynamic clustering operations on the non-recursive BSK-cube and obtains the number B_s of bus-steps required for executing them to show the potential of the network. The routing method described in Definition 2.3 is assumed for the all operations. Note that the number of bus-steps for the BSK^γ-cube $(\gamma>1)$ to execute each of the operations is equal to γB_s.

The BSK-cube cannot generally perform a permutation without bus contention. It is well known that a special type of permutation, the *simultaneous distance-d exchange* operation, is effectively used in a large number of parallel algorithms [17]. In the operation on the 2^n-node system, 2^{n-1} pairs, each of a node $\langle s,\ell\rangle$ and its partner $\langle s',\ell'\rangle$ whose addresses are different from each other at specified d bit-positions, exchange data at the same time.

For a node $\langle s,\ell\rangle$, let S-members be the nodes $\langle s,*\rangle$ in suit S_s, P-members $\langle *,\ell\rangle$ be the members of partition $P_{\langle s,\ell\rangle}$, and B-members be S-members $\cup$ P-members; so the B-members are the nodes that are connected to the bus of node $\langle s,\ell\rangle$. Then,

Theorem 3.1. *On the $BSK_n(v,w)$, the simultaneous distance-d exchange operation is performed in 1 bus-step with no bus congestion for any d $(1\le d\le n)$ and its positions.*

Proof. Depending on the positions of d bits, the partner $\langle s',\ell'\rangle$ of a node $\langle s,\ell\rangle$ is its P-member if $s\ne s'$ and $\ell=\ell'$ or its S-member if $s=s'$ and $\ell\ne\ell'$. In both cases, the datum is sent via the partner's bus. Otherwise ($s\ne s'$ and $\ell\ne\ell'$), it is sent along the bus of node $\langle s',\ell\rangle$, that is a P-member of node $\langle s,\ell\rangle$. So in the all cases and for any d and its positions, the datum is sent through the bus of the P- or S-member (referred to as the agent) of node $\langle s,\ell\rangle$. Because the S-members and the P-members of a node are connected to the node's bus, this operation is performed in one bus-step. No bus congestion is induced since the agents, and hence, their buses are different for different nodes. □

Assume a sequence $(\langle s_{0(j)},\ell_j\rangle,\langle s_{1(j)},\ell_j\rangle,\ldots,\langle s_{2^v-1(j)},\ell_j\rangle)$ in each P-members $\langle *,\ell_j\rangle$, where $s_{i(j)}\in\{0,1,\ldots,2^v-1\}$ ($s_{p(j)}\ne s_{q(j)}$ if $p\ne q$) depends on j and $\ell_j\in\{0,1,\ldots,2^w-1\}$. Let $(\langle *,\ell_0\rangle,\langle *,\ell_1\rangle,\ldots,\langle *,\ell_{2^w-1}\rangle)$ be an order of the sets of P-members. Suppose that the nodes $\langle s_{i(j)},\ell_j\rangle$ $(i\ne 2^v-1)$ send their packets to nodes $\langle s_{i+1(j)},\ell_j\rangle$, but also nodes $\langle s_{2^v-1(j)},\ell_j\rangle$ at the last positions of the sequences send the data to the nodes $\langle s_{0(j+1)},\ell_{j+1}\rangle$ at the first positions of the next sequences, all simultaneously. We call this operation *data*

streaming on the network, that will be effectively used for the streaming of, for instance, multimedia data [18] among the nodes. Then,

Theorem 3.2. *The data streaming operation is performed in one bus-step with no bus congestion on the $BSK_n(v, w)$-cube.*

Proof. The data are sent via the buses of receiving nodes, i.e., nodes $\langle s_{i+1(j)}, \ell_j \rangle$ if the streaming is within each P-members $\langle *, \ell_j \rangle$ or nodes $\langle s_{0(j+1)}, \ell_{j+1} \rangle$ otherwise. Since the receiving nodes are different from each other, the streaming completes in one bus-step with no bus congestion. □

Last, we present an operation for supporting memory systems. Suppose that a CMP consists of a number of on-chip processing nodes and off-chip memory, and that the processing node has a processor, a level-1 (L1) cache, and a level-2 (L2) cache. Then one of the problems against the CMP is how to reduce not only the number of long-distance memory requests but also the traffic to off-chip memory. A memory hierarchy will be effective if it can confine most memory requests in a local cluster where the requesting node is located. Memory requests satisfied in the local clusters can alleviate the long-wire problem, while a reduced traffic to off-chip memory can mitigate the pin-neck problem.

A memory hierarchy of private L1 and L2 caches and memory could be organized with the static clusters that are fixed in hardware. However, if the rate of memory requests is high in the cluster, the traffic to the per-cluster units, such as the directory for cache coherence and network interface, increases, leading to a large latency for the requests.

Let M_s denote the memory unit consisting of memory blocks B_{s*} whose addresses $s*$ equal s in the upper v bits. We associate one off-chip shared memory unit M_s with suit S_s to save the pin count required for the memory interface, but also combine the requests for memory units (as described below) to reduce the traffic crossing the chip boundary. Moreover, let the L2 caches in each cluster be shared in the cluster to localize the requests and to reduce the number of requests that have to cross the chip boundary.

To reduce the traffic to per-cluster units, a dynamic clustering will be effective. *Dynamic clustering* refers to the dynamic partitioning only of the nodes requesting for, as applied to the memory system, a specific memory block. So the size of a cluster is no more than the partition size 2^v, i.e., the number of nodes in the P-members. We refer to the leader ℓ of a partition P_ℓ in which a cluster is produced also as the leader ℓ of the cluster C_ℓ.

When no copy of a memory block $B_{t\alpha}$ is in the L2 cache of a node $\langle s, \ell \rangle$, we organize a set of clusters for block $B_{t\alpha}$ as follows:

Definition 3.3. In the dynamic clustering of the requests for a memory block $B_{t\alpha}$, a node $\langle s, \ell \rangle$ sends the request to the L2 cache of the leader $\langle t, \ell \rangle$ in suit S_t. The L2 cache returns the copy of block $B_{t\alpha}$ if it has the copy, or produces a single request for the sake of the requests for block $B_{t\alpha}$ received in the cluster and sends it to the memory unit M_t crossing the chip boundary otherwise.

Block $B_{t\alpha}$ read from unit M_t returns to node $\langle s, \ell \rangle$ via the reverse path. The next theorem shows the effect of the dynamic clusters, as compared with the static clusters.

Theorem 3.4. *The traffic to the leaders of clusters for memory requests is smaller than the traffic to the per-cluster units in the static clusters by a factor of at most* 2^v.

Proof. Assume that the memory requests are uniformly distributed for all memory units. Then in the static clusters, the requests not satisfied in the L2 cache concentrate on a single per-cluster unit. On the other hand, with the dynamic clusters, the requests are sent to one of the 2^v leaders depending on the requested memory-block address, so that the traffic to one leader reduces to one 2^vth of the traffic to the per-cluster unit. □

4 Conclusions

We have presented the BSK-cube and its recursive version, BSK$^\gamma$-cube, to alleviate the long-wire and pin-neck problems against high-performance NoCs. The topology of the BSK-cube is a semi-complete (SK) graph derived from the relationship between the suits of codewords and the partitions produced with the all suits; the suits and partitions are obtained using the extended Hamming code [3]. Simple partitions are introduced to increase the performance and design flexibility, but also to decrease the layout complexity of the BSK-cube.

The 2^n-node SK graph, $\mathrm{SK}_n(v, w)$, consists of 2^v number of 2^w-node complete graphs, K_w's, and each node in every K_w is connected to a node in each of the other K_w graphs; then the 2^v nodes in different K_w graphs organize another complete graph K_v. For the SK_n graph with a large n, we have recursively convert its K_w components into $\mathrm{SK}_{n'}$ graphs with a smaller size n' to obtain the γ-level recursive SK_n graph, $\mathrm{SK}_n^\gamma(v_1, \ldots, v_\gamma, w_\gamma)$, where $\sum_{i=1}^{\gamma} v_i + w_\gamma = n$. The BSK_n^γ-cube is produced from the SK_n^γ by replacing the links in each constituent SK that are incident to a node by a single bus for the node.

The BSK_n^γ is laid out on a $2^{v_1} \times 2^{v_2}$ array of $\mathrm{BSK}_n^{\gamma-2}$ cubes and has γ buses per node. The diameter equals γ. The buses of a node run along the same row and the same column as the node is located, so that the buses have very regular wiring patterns. The bus length is $\mathbf{O}(2^{n/2})$ and $\mathbf{O}(2^{v_\gamma})$ for the level-1 and level-γ buses, respectively. We can mitigate the long-wire problem if we can map computations so that they are almost locally performed since dominant communications are then performed via the short length level-γ bus in one bus-step, or if we implement the lower levels of buses with larger feature sizes.

We have shown by analysis that simultaneous distance-d exchange ($1 \le d \le n$) and data streaming operations are performed in γ bus-steps with no bus congestion. Another operation, dynamic clustering, reduces the traffic to the per-cluster units such as the cache coherence directory by a factor of up to 2^v as compared with static clustering. This will reduce the delay for off-chip memory requests, as well as alleviate the pin-neck problem.

We are evaluating the effects of dynamic clustering of memory requests; the results will be presented in a future paper. Streaming of video data in a memory hierarchy is also attractive; we are developing a cache protocol to support data streaming.

Acknowledgments

We thank the anonymous reviewers for the constructive comments for refining the paper.

A A Hamming code-based BSK-Cube

This appendix shows an example of the BSK-cube, BSK_6-cube, that is organized from the Hamming code-based partitions (see Fig. A1).

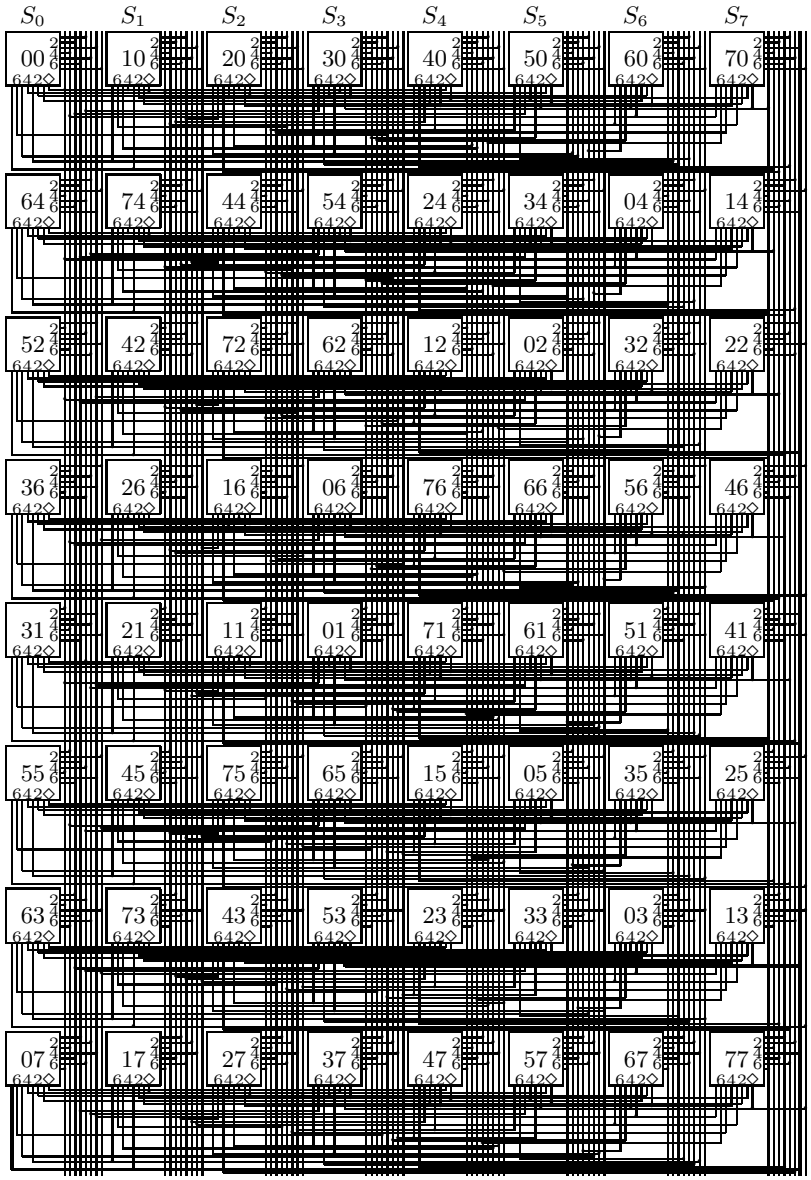


Fig. A1. The BSK_6-cube organized from the Hamming code-based partitions.

Let p and k be the sizes of parity and information parts, and ρ and ι denote the values of the parts. Then nodes of the BSK_6-cube are arranged into the $2^k \times 2^p$ array ($k = p = 3$

when $n = 6$). The node address $\langle \rho, \iota \rangle$ is shown by $\rho\iota$ for space in each node box. The nodes in each suit S_c ($0 \leq c \leq 7$) are mapped on the positions with indices $(\mathrm{PS}^{-1}(k, \iota), c)$; for the perfect shuffle (PS) code, see [5]. For instance, node $\langle 1, 3 \rangle \in S_7$ is mapped on the position of index $(r, c) = (6, 7)$ since $r = \mathrm{PS}^{-1}(k, \iota) = \mathrm{PS}^{-1}(3, 3) = 6$ and $c = 7$.

The bus of node $\langle \rho, \iota \rangle$ on the position with index (r, c) is connected to all nodes in column c (i.e., in S_c) and to the members of partition $P_{\langle \rho, \iota \rangle}$ that are distributed to 4 rows. The ports $(0, 1, \ldots, 7)$ (port 0 is symbolized by $\diamond$) of node $\langle \rho, \iota \rangle$ for connecting the members of $P_{\langle \rho, \iota \rangle}$ are shown at the bottom in the node, and those for connecting the nodes in $S_{\ni \langle \rho, \iota \rangle}$ are along the right side in the node.

The patterns of bus wiring are more complex than those in the layout of the BSK_6 based on the simple partitions (see Fig. 5), since the members of a partition have to be mapped on $k+1 = 4$ rows when $n = 6$. The wiring is, however, regular in the sense that the patterns along the rows and those along the columns are the same, respectively. The bus length is $\mathbf{O}(2^p + 2^k)$ that happens to be equal to the length in the layout shown in Fig. 5 since $n = 6$; in general, this is not the case. Over all, the properties of the BSK-cubes obtained with the Hamming code-based and the simple partitions are almost the same, except of the layout complexity, and hence, the bus length; this leads to a performance difference.

Let $n_0 = n$ and $n_i = k_{i-1}$ $(= p_i + k_i)$. Then the BSK^γ ($\gamma > 1$) is organized by the recursive partitioning of the nodes in each suit in the n_{i-1}-bit space because the information parts of their addresses cover the n_i-bit space. The diameter equals γ, and the bus length is $\mathbf{O}(2^L)$ in the level 1, where $L = \sum_{i=1}^{\gamma} \ell_i$ and ℓ_i is p_i or k_i that makes L smallest.

References

[1] D. Matzke, "Will Physical Scalability Sabotage Performance Gains?," IEEE Computer, Vol. 30, No. 9, Sep. 1997, pp. 37-39.

[2] J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Qualitative Approach," Third Edition, Morgan Kaufmann Publishers, 2003.

[3] M. Takesue, "Ψ-Cubes: Recursive Bused Fat-Hypercubes for Multilevel Snoopy Caches," Proc. Int. Symp. on Parallel Architectures, Algorithms, and Networks (I-SPAN), June 1999, pp. 62-67.

[4] M. Takesue, "DC-Mesh: A Contracted High-Dimensional Mesh for Dynamic Clustering," Proc. 2004 IFIP Int. Conf. on Network and Parallel Computing, Springer LNCS 3222, Oct. 2004, pp. 382-389.

[5] M. Takesue, "The Psi-Cube: A Bus-Based Cube-Type Network for High-Performance On-Chip Networks," Proc. 2005 Int. Conf. on Parallel Processing (ICPP) Workshops, pp. 539-546, June 2005.

[6] T. Lovett and R. Clapp, "STiNG: A ccNUMA Computer System for the Commercial Marketplace," Proc. 23th Int. Symp. on Computer Architectures, May 1996, pp. 308-317.

[7] J. Laudon and D. Lenoski, "The SGI Origin: A ccNUMA Highly Scalable Server," Proc. 24th Int. Symp. on Computer Architectures, Jun. 1997, pp. 241-251.

[8] K. Olukotun et al., "The Case for a Single Chip Multiprocessor," Proc. 7th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, 1996, pp. 2-11.

[9] L. A. Barroso et al, "Piranha: A Scalable Architecture Based on Single-Chip Multiprocessors," Proc. 27th Int. Symp. on Computer Architectures, Jun. 2000, pp. 282-293.

[10] K. Sankaralingam et al., "Exploiting ILP, TLP, and DLP with the Polymorphous Architecture," Proc. 30th Int. Symp. on Computer Architectures, June 2003, pp. 422-433.

[11] P. Guerrier and A. Greiner, "A Generic Architecture for On-Chip Packet-Switched Interconnections," Proc. Design and Test in Europe (DATE), pp. 250-256, Mar. 2000.

[12] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," Proc. Design Automation Conf. (DAC), pp. 683-689, 2001.

[13] F. Karim et al., "An Interconnect Architecture for Networking Systems on Chips," IEEE Micro, Vol. 22, No. 5, pp. 36-45, Sep./Oct. 2002.

[14] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnection in Multi-Core Architectures: Understanding Mechanisms, Overhead and Scaling," Proc. 32nd Int. Symp. on Computer Architectures, pp. 408-419, June 2005.

[15] S. B. Akers and B. Krishnamurthy, "A Group-Theoretic Model for Symmetric Interconnection Networks," IEEE Trans. on Computer, Vol. C-38, No. 4, Apr. 1989, pp. 555-566.

[16] W.-J. Hsu, M. J. Chung, and A. Das, "Linear Recursive Networks and Their Application in Distributed Systems," IEEE Trans. on Parallel and Distributed Systems Vol. 8, No. 7, July 1997, pp. 673-680.

[17] V. Kumar, A. Grama, A. Gupta, and G. Karypis, "Introduction to Parallel Computing: Design and Analysis of Algorithms," Reading, Benjamin/Cummings Publishing Company, Inc., 1994.

[18] U. J. Kapasi, S. Rixner, W. J. Dally, B. Kbailany, J. H. Abn, P. Mattson, and J. H. Owens, "Programmable Stream Processors," IEEE Computer, Vol. 36, No. 8, Aug. 2003, pp. 54-62.

Masaru Takesue
Dept. Electronics and Information Engr.
Hosei University, Tokyo 184-8584 Japan
takesue@ami.ei.hosei.ac.jp

On the numerical simulation of nonlinear integrate-and-fire neurons

Arnaud Tonnelier

Abstract

Studies in computational neuroscience rely on the simulation of neural models. We review here some methods for the numerical integration of integrate-and-fire neuron models and we present a new integration scheme based on a voltage-stepping approximation. We illustrate this method on the numerical simulation of quadratic integrate-and-fire model.

Introduction

Neurons communicate by spikes and there is an increasing number of experiments that show that spike timing contains most of the relevant information. In some behavioral experiments, reaction times are very short so that only one spike from each neuron is involved per processing step [15]. Synaptic plasticity depends on the relative timing of postsynaptic and presynaptic spikes [11]. The resulting change is thought to be involved in learning processes. The theoretical understanding of the dynamics of neuronal systems needs the simulation of spiking neurons. Biophysically realistic spiking neuron models do not allow for analytical studies and are highly time consuming. Simplified models of integrate-and-fire (IF) type have retained an increasing attention [5]. In IF models, the spike event is reduced to a discrete event, the so-called firing time. Two simulation strategies have been developed. Exact methods where spike time is calculated exactly can be applied to a restricted class of non realistic neural models. Time-stepping methods can be applied to any model but do not allow for a reliable approximation of the firing times. We review both and propose an hybrid method based on a mixture of exact and numerical calculations that overcomes the main disadvantages of previous integration schemes.

Estimation of the firing time is crucial and fixed time-stepping methods do not provide an accurate approximation when the firing time is estimated at the boundaries of the time steps. Adaptive time-stepping methods are required for an accurate determination of the firing times. We show that an implicit adaptive time-stepping method could be defined using a voltage-stepping scheme. This numerical method is well suited for the simulation of IF models.

1 Exact method

One of the simplest model that describes spiking neurons is the leaky IF model [8], [16]. Only the passive membrane properties are described and the membrane potential evolves below the firing threshold ϑ according to

$$C\frac{dv}{dt} = g_l(v_l - v) + I_0 + I_{\rm syn}(t) \quad (1)$$

where g_l is the conductance, v_l the reversal potential, I_0 is a constant external current and I_{syn} is a synaptic current due to the action of other neurons. The membrane capacitance C is set at $1\mu F cm^{-2}$. The form of an action potential is not described explicitly and spikes are formal events characterized by a firing time t^f defined by a threshold criterion

$$v(t^f) = \vartheta \quad (2)$$

Immediately after t^f, the potential is reset to a subthreshold value $v_r < \vartheta$,

$$\lim_{t\to t^f, t>t^f} v(t) = v_r \quad (3)$$

The neural model (1)-(3) is a differential equation with state-dependent impulses. Such equations were considered at the beginning in nonlinear mechanics [14] and have attracted in recent years the attention of the computational neuroscience community.
Since the subthreshold regime is linear eq. (1) can be integrated exactly. Let us suppose that a spike has occurred at $\hat{t}$. The reset value v_r can be treated as an initial condition and the integration of (1) gives for $t > \hat{t}$

$$v(t) = v_r e^{-g_l(t-\hat{t})} + (I_0 + g_l v_l)(1 - e^{-g_l(t-\hat{t}}) + \int_{\hat{t}}^{t} e^{-g_l(t-y)} I_{syn}(y)dy \quad (4)$$

up to the moment of the next threshold crossing. The synaptic currents are generated by the presynaptic spikes t^f_{pre} and the total synaptic current is given by

$$I_{syn} = g_{syn} \sum_{t^f_{pre}} \alpha(t - t^f_{pre})$$

where

$$\alpha(t) = \frac{1}{\tau_1 - \tau_2}(e^{-\frac{t}{\tau_1}} - e^{-\frac{t}{\tau_2}}) \quad (5)$$

for $t > 0$ and 0 otherwise. Parameters τ_1 and τ_2 are the synaptic time constants. The integral in (4) can be evaluated analytically and the time at which the neurons fires is obtained as the solution of a transcendental equation. Event-driven simulations with the synaptic currents (5) or instantaneous interactions have been developed [10], [13]. More realistic currents, i.e. synaptic conductances, have been used leading to exact calculations

but involving tricky calculations [1].
It is known that leaky IF models do not correctly reproduce the neuronal dynamics close to the firing threshold. Nonlinear IF models provide a more accurate description of the neural activity. The membrane potential of nonlinear IF models evolves according to the equation:

$$C\frac{dv}{dt} = f(v) + I_0 + I_{\text{syn}}(t) \tag{6}$$

where f is a nonlinear function that models the spike-generating current. Special cases are the quadratic IF model where $f(v) = g_l(v_l - v)^2$ [3], or the exponential IF model where $f(v) = g_l e^{v-v_l}$ [4]. The strict voltage threshold (2) is replaced by a more realistic smooth spike initiation. In general it is no longer possible to calculate analytically the membrane potential and exact simulation methods can not be extended to non linear IF. For this reason, approximation methods using steps are required to simulate realistic neuron models.

2 Time-stepping Schemes

Time-stepping methods such as Euler, Runge-Kutta or Crank-Nicolson are the most popular numerical methods. They have been extensively used for the numerical integration of nonlinear neuronal models [12]. However the IF dynamics present discontinuities of the membrane potential which may cause some numerical problems. These methods fail to be efficient when accuracy is required on the firing time and not on the membrane potential. A naive implementation where the spike is assumed to be fired at a fixed time step leads to a global error that is dominated by the error due to the discontinuity of the dynamics limiting the algorithm to first order in time. Modified time-stepping methods can increase the accuracy of the numerical integration leading to second-order schemes [7], [9]. Finding explicit adaptive time steps can be sophisticated leading to algorithms where the accuracy and efficiency are difficult to assess.

3 Voltage-stepping Schemes

The neural modeling involves the coexistence of continuous differential equations with discrete events and methods coming from hybrid systems theory [2], [6] could improve the numerical integration of neural models. An integration method based on a voltage discretization seems to be well suited to capture the non smooth nature of the neural dynamics. Let us consider that the membrane potential is in the interval $v \in V_k = [k\Delta v, (k+1)\Delta v]$ where Δv be the voltage step. In a voltage-stepping method the non linear part of the differential equation, $f(v)$, is approximated on V_k by a linear function yielding the differential equation: $C\frac{dv}{dt} = g_l^k(v_l^k - v) + I_0 + I_{\text{syn}}(t)$ for $v \in V_k$. When v reaches the threshold interval a spike is fired and v restarts in the resetting interval. Locally, the non linear IF is approximated by a leaky IF models that allows for the use of the exact methods previously developed. The voltage discretization implicitly defines a corresponding time discretization

where the time steps are the times at which v leaves the intervals (V_k). The major advantage is to define an implicit time step leading to short time steps when the membrane potential varies quickly. This method is reminiscent to the piecewise linear reduction of smooth neural models [17].
Let us illustrate this numerical scheme using the quadratic integrate-and-fire (QIF) model. The QIF is a canonical type I neural model which is widely used to model the neural activity. For $f(v) = g_l(v_l - v)^2$ we choose the linear approximation on V_k given by the interpolation at $k\Delta v$ and $(k+1)\Delta v$. It is straightforward to calculate

$$\begin{aligned} g_l^k &= g_l(2v_l - (2k+1)\Delta v), \\ v_l^k &= (v_l^2 - k(k+1)\Delta v^2)/(2v_l - (2k+1)\Delta v). \end{aligned}$$

For clarity we take $I_{syn} = 0$ and we consider the neuron under a positive constant external drive that makes the neurons fire periodically. One of the fundamental properties of neuron is the input-output transformation classically characterized by its frequency current (f-I) relationship. The f-I curve of the QIF can be computed analytically yielding to a precise evaluation of the accuracy of the numerical integration method. The frequency of the QIF under an external drive $I_0 > 0$ is

$$f = \frac{\sqrt{g_l I_0}}{\arctan \frac{\sqrt{g_l}(\vartheta - v_l)}{\sqrt{I_0}} - \arctan \frac{\sqrt{g_l}(v_r - v_l)}{\sqrt{I_0}}} \tag{7}$$

Moreover, we compare the voltage-stepping method with the corresponding time-stepping scheme i.e. the standard Euler algorithm where the membrane potential is calculated according to $v(t + \Delta t) = v(t) + \Delta t(g_l(v_l - v)^2 + I_0)$ where Δt is the time step. When $v(t + \Delta t) > \vartheta$, a spike is assumed to be fired at time $t + \Delta t$ and the membrane potential is reset to v_r.
In our simulations we take a leak conductance $g_l = 0.1mS/cm^2$ that leads to a passive membrane time constant $\tau = C/g_l = 10ms$. Other values are $v_l = -60mV$, $v_r = -70mV$ and $\vartheta = -40mV$. Figure 1 displays the f-I curve of the QIF computed with the exact expression (7), the Euler method and the proposed voltage-stepping scheme using two different step sizes for both Δt and Δv. A small step size is required to accurately reproduce the f-I curve using an Euler method. Numerical artifacts due to the discontinuity of the dynamic appear using the time-stepping method. The voltage-stepping scheme gives a good accuracy and the convergence is very fast. Note that the time step from fig.1 A to fig.1 B is divided by 10 whereas the voltage step is only divided by 2.

4 Discussion

It is known that standard integration algorithms give reliable results only for very small integration steps. To improve the integration efficiency adaptive time-stepping methods are desirable. We have shown that a voltage-stepping scheme defines an implicit adaptive time-stepping method that greatly improves the numerical integration.

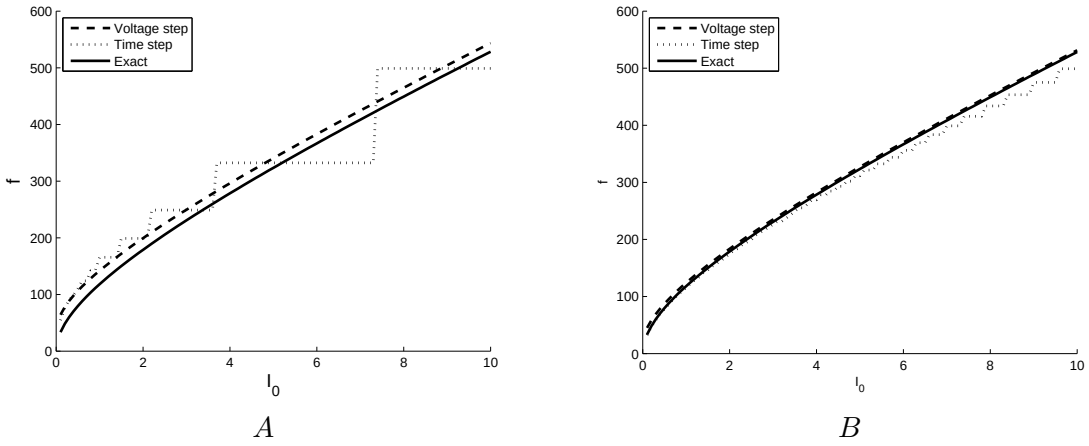


Figure 1: Graph of the f-I curve of the quadratic integrate-and-fire neuron obtained with the voltage-stepping method (dashed line), a time-stepping scheme (dotted line) and the exact value (solid line). Parameters are (A) $\Delta v = 5mV$, $\Delta t = 1ms$ and (B) $\Delta v = 2.5mV$, $\Delta t = 0.1ms$.

We evaluate the numerical accuracy of the voltage step method using a constant input scenario. More realistic inputs, random current injection and conductance injection, have to be used to analyze more closely the efficiency of the method. Finally, efforts should be made to extend the method to the simulation of networks of interacting neurons.
In other fields of science similar numerical simulation issues have occurred: when to use time-stepping or event-driven schemes? We expect that time-stepping methods will be efficient when the number of events is high, i.e. in network with high conductance state dynamics, whereas event-based simulations will be better at low firing rate activity.

References

[1] R. Brette. Exact simulation of integrate-and-fire models with synaptic conductances, *submitted*. 2005.

[2] J. Della Dora, A. Maignan, M. Mirica-Ruse and S. Yovine. Hybrid computation. *Proc. International Symposium of Symbolic and Algebraic Computation*, 691-704, 2001.

[3] B. Ermentrout. Type I, phase resetting curves, and synchrony. *Neural Comp. 8*, 979-1001, 1996.

[4] N. Fourcaud-Tromé, D. Hansel, C. van Vreeswijk and N. Brunel. How spike generation mechanisms determine the neuronal response to fluctuating input. *J. Neurosci 23*, 11628-11640, 2003.

[5] W. Gerstner and W.M. Kistler. Spiking neuron models. Cambridge University Press, 2002.

[6] A. Girard. Analyse algorithmique des systèmes hybrides. Thèse de doctorat, INP Grenoble, 2004.

[7] D. Hansel, G. Mato, C. Meunier and L. Neltner. On the numerical simulations of integrate-and-fire neural networks. *Neural Comput. 10*, 2, 1998.

[8] L. Lapicque. Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *J. Physiol. Pathol. Gen. 9*,620-635, 1907.

[9] W.W. Lytton. Independent variable time-step integration of individual neurons for network simulations. *Neural Comput. 17*, 903-921, 2005.

[10] T. Makino. A discrete-event neural network simulator for general neuron models. *Neural. Comput. and Applic. 11*, 210-223, 2005.

[11] H. Markram, J. Lubke, M. Frotscher and B. Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science 275*, 213-215, 1997.

[12] M.V. Mascagni and A. Sherman. Numerical methods for neuronal modeling. In Methods in Neuronal Modeling. From Ions to Networks, C. Koch, and I. Segev, second edition. MIT Press, Cambridge, MA. 1998.

[13] M. Mattia and P. Del Giudice. Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Comput. 12*, 10, 2000.

[14] A.M. Samoilenko and N.A. Prerstyuk, Impulsive differential equations. World scientific series on nonlinear science, Series A, vol. 14, 1995.

[15] S. Thorpe, D. Fize and C. Marlot. Speed of processing in the human visual system. *Nature 381*, 520-522, 1996.

[16] H.C. Tuckwell. Introduction to theoretical neurobiology. Cambridge University Press, Cambridge, 1988.

[17] A. Tonnelier and W. Gerstner. Piecewise linear differential equations and integrate-and-fire neurons : Insigth from two dimensional membrane models. *Phys. Rev. E 67*, 2004.

A. Tonnelier
LORIA & LMC-IMAG
Arnaud.Tonnelier@loria.fr
www.loria.fr/~tonnelia

Markov Chains, Iterated System of Functions and Coupling time for Perfect Simulation *

Jean-Marc Vincent

Abstract

Simulation of Markov chains are usually based on an algorithmic representation of the chain. This corresponds to stochastic recurrent equation and could be interpreted as random iterated systems of functions (RIFS). In particular, for perfect simulation of Markov chains, the RIFS structure has a deep impact on execution time of the simulation. Links between the structure of the RIFS and coupling time of algorithm are detailed in this paper. Conditions for coupling and upper bound for simulation time are given for Doeblin matrices. Finally, it is shown that aliasing techniques build an RIFS with a particular binary structure.

1 Introduction

Markov chains are basic tools to study random dynamical systems. They play the central role of linear part of the dynamic and capture most of the dynamic characteristics. When the system is finite, the Markov chain is described by its transition kernel (stochastic matrix). When the system is homogeneous in time, irreducible and aperiodic, the left eigenvector π associated to the eigenvalue 1 captures most of informations needed in practical applications. Difficulties arise when the size of the system is too large so that traditional linear algebra tools could not be used.

For a large state space, simulation provides methods based on an algorithmic representation of the chain and offers new possibilities for the statistical estimation of π. Unfortunately, these methods are empirical and the management of errors is very difficult. Perfect simulation techniques [9] have been developed in the last 10 years. These methods guarantee the convergence to steady-state in a finite number of steps and help for the simulation control.

In fact, the algorithmic representation of the Markov chain could be interpreted as a random iterated systems of functions. The aim of this article is to investigate relations between the Markov chain and its representations as RISF. It is shown that the RISF impacts deeply the simulation time for perfect simulation.

In the second section the RISF formalism is introduced and properties of Markov chains from a RISF are deduced. Then forward simulation and its drawbacks is presented. Section

*This work was partially supported by ACI SurePath and ANR SMS

5 introduces the perfect simulation algorithm and the convergence criteria in a finite number of steps. Relations between the RISF and coupling time are given in section 6 and applied for Doeblin matrice (section 7). Finally the alias technique is described and the uniform-binary decomposition is proposed.

2 Random Iterated System of Functions

Consider a finite state space identified to the set $\mathcal{X} = \{1, \cdots, K\}$ and consider a finite family of m functions $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$ that maps $\mathcal{X}$ onto itself. Denote by F_θ the matrix of the operator f_θ with K rows and m columns,

$$F_\theta(i,j) = \begin{cases} 1 & \text{if } f_\theta(i) = j; \\ 0 & \text{if not.} \end{cases}$$

A probability distribution $\mathcal{P} = \{p_\theta : \theta \in \Theta\}$ is given on the set Θ of functions and the dynamic of the system is defined by

$$X_0 = x_o,\ X_1 = f_{\theta_1}(X_1),$$

$$X_{n+1} = f_{\theta_{n+1}}(X_n) = f_{\theta_{n+1}} \circ f_{\theta_n} \circ \cdots f_{\theta_1}(X_0), \tag{1}$$

where $\{\theta_n\}_{n\in\mathbb{N}}$ is a random sequence of elements of Θ chosen independently according to distribution $\mathcal{P}$. It is clear that the stochastic process $\{X_n\}_{n\in\mathbb{N}}$ is a homogeneous discrete time Markov chain, because the conditional distribution of the future does not depend on the past.

In the domain of probability such stochastic recurrence equations have been widely studied when the state space is continuous [5] or from an ergodicity point of view [2, 1, 10]. More general results are developed in [8] and [4].

Because the state space is finite, the dynamic of the process is given by the transition matrix P of the Markov chain. It is obtained by

$$p_{i,j} \stackrel{\Delta}{=} \mathbb{P}(X_{n+1} = j | X_n = i) = \sum_{\theta\in\Theta} p_\theta . \mathbb{1}_{f_\theta(i)=j}.$$

This transition matrix is non-negative and the sum of elements on a row equals 1.

The irreducibility of the Markov chain is related to pattern properties of functions.

Proposition 2.1 (Irreducibility). *Suppose that for each couple (i,j) of states there exists a finite pattern $(\theta_1, \cdots, \theta_l)$ such that the probability of the pattern is positive $p_{\theta_1} \cdots p_{\theta_l} > 0$ and*

$$j = f_{\theta_l} \circ f_{\theta_{l-1}} \circ \cdots f_{\theta_1}(i),$$

then the Markov chain associated to the RISF is irreducible.

Moreover, aperiodicity of the chain is deduced from the support of functions f_θ.

Proposition 2.2 (Aperiodicity). *The irreducible Markov chain is aperiodic if for each couple (i,j) there exist some n_0 such that for every $n \geq n_0$ there is a sequence $(\theta_1, \cdots, \theta_n)$ such that for each couple (i,j) of states,*

$$j = f_{\theta_n} \circ f_{\theta_{n-1}} \circ \cdots f_{\theta_1}(i).$$

One should note that if the Markov chain is irreducible and aperiodic there exists n_0 such that P^{n_0} is positive. Then the central convergence theorem from Kolmogorov, extension of the Perron-Froebenius in the finite case, could now be reformulated using random iterated system of functions:

Theorem 2.3 (Kolmogorov). *If the RISF is aperiodic and irreducible (recurrent positive), then there exist a unique probability measure $\pi = (\pi_1, \cdots, \pi_K)$ (line vector) satisfying*

$$\pi = \pi P = \sum_{\theta} p_\theta \pi F_\theta, \tag{2}$$

and for all (i,j)

$$\lim_{n \to +\infty} \mathbb{P}(f_{\theta_n} \circ f_{\theta_{n-1}} \circ \cdots f_{\theta_1}(i) = j) = \pi_j.$$

Intuitively, when we stop a RIFS after a long period, the probability that the observed value is j is approximatively π_j. Moreover, if we compute the proportion of steps spent state j, this proportion converges to π_j (ergodic theorem).

Theorem 2.4 (Ergodic theorem). *When the RISF is recurrent positive, then the Cesaro limit converges :*

$$\lim_{n \to +\infty} \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}_{f_{\theta_n} \circ f_{\theta_{n-1}} \circ \cdots f_{\theta_1}(i) = j} = \pi_j \text{ almost surely.}$$

Theorem 2.3 guarantees that sampling independent sufficiently long trajectories gives an estimate of the stationary distribution. The ergodic theorem 2.4allows sampling on only on a single trajectory because the probability that the Cesaro limit does not converge is 0.

3 Estimation of π

When the size of the system is sufficiently small, formal or numerical computations provide the eigenvector π of the transition matrix P. If the size of the state space is too large, a simulation builds an estimate of π. The first technique called forward simulation is based on theorem 2.3 and leads to algorithm 3.1.

Algorithm 3.1. Forward simulation (independent sample generation)

$n = 0$;
$x = x_0$;
{choice of the initial state at time n=0}

repeat
$n = n + 1$;
$f_\theta = Random\_function()$;
{Random function chosen according to the distribution $\{p_\theta : \theta \in \Theta\}$}
$x = f_\theta(x)$;
{computation of the next state X_{n+1}}
until n = simulation length
return x

This algorithm returns a state and we hope that, for a sufficiently long simulation run, the returned state distribution is a good approximation of π. So repeating the algorithm, we get a sample of independent realizations of π distributed random variables.

The problem of this approach is first the estimation of the simulation length (stabilization time or burn-in time). In usual softwares, this value is fixed empirically by the user. Moreover, because we generate a sample of independent variables the convergence of estimates of π converges very slowly to the limit value, in the order of $\mathcal{O}(\frac{1}{\sqrt{n}})$.

This algorithm could be extended by making sampling directly on the trajectory (ergodic sampling). In that case, samples are not independent and we should assume mixing properties of the process to justify the speed of convergence. Moreover, it has been shown that the convergence to steady-state depends on the spectral gap of the matrix P (module of the difference between 1 the first eigenvalue and the second eigenvalue).

Suppose now given a transition matrix P, the problem is to compute or estimate the steady state distribution π. The classical method consists in three steps :
(1) Build a set of functions $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$ and the corresponding probabilities $\mathcal{P} = \{p_\theta : \theta \in \Theta\}$ such that $P = \sum_\theta p_\theta F_\theta$;
(2) Simulate a sample by algorithm 3.1;
(3) Estimate statistics on the sample.

Proposition 3.2. *The convergence of the forward algorithm does not depend on the set of function $\mathcal{F}$ and $\mathcal{P}$.*

This is clear because the construction of $\mathcal{F}$ does not modify the matrix P and so convergence to steady state. Usually, the construction of the family of functions $\mathcal{F}$ and $\mathcal{P}$ are based on randomized algorithms. Because the set of states is finite, usual algorithms are like the following 3.3 (inverse probability distribution function):

Algorithm 3.3. Next state generation
{Current state is i}
$s = 0$; $j = 1$;
$u = Random(0, 1)$;
while $u > s$ **do**
$s = s + P[i, j]$; $j = j + 1$;
end while
return j

This representation leads to set $\mathcal{F}$ with cardinality at most $m-(K-1)$ where m is the number of positive elements of P. This algorithm could be improved, by tree structures or hash tables, the main idea is to consider an appropriate segmentation of the interval $[0,1[$ as shown in section 8.

4 Forward coupling

An intuitive idea (not so good as shown in example 1), to stop simulation is to consider all possible initial values, observe their trajectories and stop the simulation when they are all in a same state. We say that all trajectories have coupled. The coupling time is the first time when the trajectories are all in the same state, after the coupling time, the trajectories do not depend on the initial state. The recurrent equation (1) is applied to each state of $\mathcal{X}$ and we denote by $y(x)$ the current value of the trajectory issued from state x.

Algorithm 4.1. Forward-coupling simulation
for all $x \in \mathcal{X}$ **do**
$y(x) = x$;
{choice of the initial value of the vector y, $n = 0$}
end for
repeat
$n = n + 1$;
$f_\theta = Random\_function()$;
{Random function chosen according to the distribution $\{p_\theta : \theta \in \Theta\}$}
for all $x \in \mathcal{X}$ **do**
$y(x) = f_\theta(y(x))$;
{computation of the next state of the trajectory issued from x at time 0}
end for
until All $y(x)$ are equal
return $y(x)$

An example of a forward-coupling simulation is illustrated by figure 1.

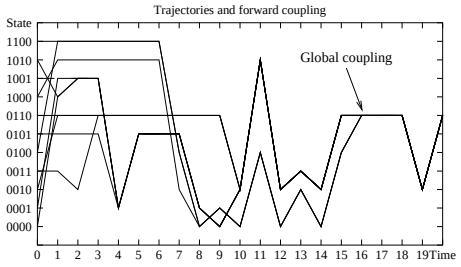
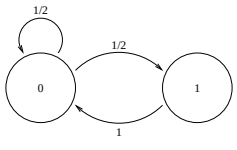


Figure 1: All trajectories have coupled before time $\tau^* = 16$

When the forward coupling algorithm stops all trajectories have coupled, unfortunately the generated state does not follows the stationary distribution. This is clearly illustrated in the example 1.

Example 1 : Coupling in a same state



On the example to the left, it is clear that coupling does not depend of the representation and that coupling time is almost surely finite, geometrically distributed with parameter $\frac{1}{2}$. When 2 trajectories couple, at the preceding step the corresponding states were 0 and 1. But, because the transition probability from 1 to 1 is zero, the trajectories can only couple in 0. Then the generated state is always 0, and is not distributed according to the stationary distribution $\pi = [\frac{2}{3}, \frac{1}{3}]$.

5 Backward Simulation Scheme

To make this algorithm "exact", [9] propose to shift the process in the past. This is equivalent to [7] monotone scheme used to prove the law convergence of the workload of a queuing system.

Provided that the representation of the Markov chain ensures coupling, we modify the algorithm (4.1) by reversing time leading to algorithm 5.1:

Algorithm 5.1. Backward-coupling simulation

for all $x \in \mathcal{X}$ **do**
  $y(x) \leftarrow x$ {choice of the initial value of the vector y, $n = 0$}
end for
repeat
  u $\leftarrow$ Random; {generation of $f_{\theta_{-n}}$}
  for all $x \in \mathcal{X}$ **do**
    $y(x) \leftarrow y(f_{\theta_{-n}}(x))$; {computation of the state at time 0 of the trajectory issued from x at time $-n$}
  end for
until All $y(x)$ are equal
return $y(x)$

We illustrate the behavior in figure (2).

To understand this algorithm and find conditions for termination, we consider the sequence of subsets of the state space $\mathcal{X}$, $\{\mathcal{Z}_n\}_{n\in\mathbb{N}}$ defined by

$$\mathcal{Z}_n = f_{\theta_{-1}} \circ f_{\theta_{-2}} \circ \cdots f_{\theta_{-n}}(\mathcal{X}). \tag{3}$$

Because $f_{\theta_{-n}}(\mathcal{X}) \subset \mathcal{X}$, we deduce that the sequence $\{\mathcal{Z}_n\}_{n\in\mathbb{N}}$ is non-increasing. Using the finiteness of $\mathcal{X}$ and monotonicity, we obtain that $\{\mathcal{Z}_n\}_{n\in\mathbb{N}}$ converges almost surely to a set $\mathcal{Z}_\infty$. The system is coupling if $\mathcal{Z}_\infty$ is reduced to one point.

The next theorem, [9, 12], states the fundamental result of the method:

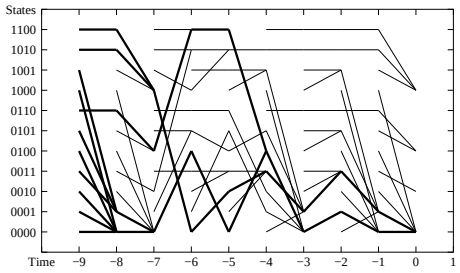


Figure 2: All trajectories collapsed in state 0000 after 9 steps

Theorem 5.2. *Suppose that the algorithm terminates, then the generated value $y(x)$ by algorithm 5.1 is distributed according to the stationary distribution.*

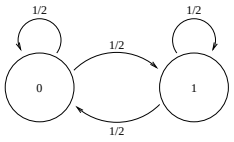
The difficulty is now to obtain conditions under which the coupling time is finite almost surely. An example which illustrate this difficulty is given in section 6 and construction of RIFS ensuring that the algorithm terminates are detailed in section 7 and 8.

6 Coupling time

It was shown in proposition3.2 that the choice of the RIFS implementing the Markov chain did not affect the convergence of the process. In the backward coupling scheme the situation is clearly different. Consider the following example of a two states Markov chain, in which we suppose that the$\{U_n\}$ are uniformly distributed on $[0, 1]$.

Example 2 : Three representations of a same Markov chain

This example, as example 1 before, is derived from the course of [6, chap. 10]. Consider the simple two states Markov chain :



Its transition matrix is

$$P = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix},$$

the chain has a unique stationary distribution

$$\pi = (\frac{1}{2}, \frac{1}{2}).$$

The three following RISF $f = (f_1, f_2)$ with probability $p_f = (\frac{1}{2}, \frac{1}{2})$, $g = (g_1, g_2)$ with probability $p_g = (\frac{1}{2}, \frac{1}{2})$, and $h = (h_1, h_2, h_3, h_4)$ with probability $p_h = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ whose values are given by

$$\begin{cases} f_1(0) = f_1(1) = 0, \\ f_2(0) = f_2(1) = 1; \end{cases} \quad \begin{cases} g_1(0) = g_2(1) = 0, \\ g_1(1) = g_2(0) = 1; \end{cases} \quad \text{and}$$

$$\begin{cases} h_1(0) = h_1(1) = h_2(0) = h_3(1) = 0, \\ h_2(0) = h_3(0) = h_4(0) = h_4(1) = 1, \end{cases}$$

represent the same transition matrix P. The coupling time for the first representation equals 1, for the second representation the algorithm never terminates and the coupling time is geometrically distributed with mean 2 for the last case.

This example shows that the construction of the RISF is of crucial importance on the coupling time. We observe that the behavior of the system could exhibit mean coupling time arbitrarily large and in some cases the coupling time is infinite.

7 Doeblin matrices

Consequently, to avoid this problem, the idea is to build the RISF in order to ensure coupling. Constant functions are of great interest as :

Proposition 7.1. *If there exist a constant function$f_\theta \in \mathcal{F}$ then the coupling time is almost surely finite and stochastically bounded above by a geometric distribution with parameter p_θ.*

When the function f_θ is picked, all trajectories collapse in one state and coupling occurs. The coupling time is dominated by the first occurrence time of f_θ in the independent process of $\{f_{\theta_n}\}$ which is geometrically distributed.

When the transition matrix is positive (Doeblin matrix), it is possible to build a RIFS which could couple in each state in just one step. Denote by

$$\alpha_j = \min_i p_{i,j}.$$

The Doeblin condition is simply

$$\min_j \alpha_j > 0. \tag{4}$$

Consequently the the algorithm 7.2 compute the next step of the chain and ensures coupling.

Algorithm 7.2. Next states generation (Doeblin matrices)

$u = Random(0, 1)$;
$s = 0$; $j = 1$;
while $u > s$ **do**
  $s = s + \alpha[j]$, $j = j + 1$
end while
for $i = 1$ to n **do**
  $t = s$; $j = 1$;
  while $u > s$ **do**
    $s = s + P[i, j] - \alpha[j]$; $j = j + 1$;
  end while
  $NextState(i) = j$
end for

Proposition 7.3. *The RIFS represented by algorithm 7.2 couples in finite time and the mean coupling time is less than*

$$\frac{1}{\sum_{i=1}^{K} \alpha_i}$$

In fact this proposition is still valid and the algorithm still works if there is a column of positive elements, so inequality 4 is replaced by the weaker inequality:

$$\max_i \left\{ \min_j p_{i,j} \right\} > 0.$$

This condition implies that there exists a state that is accessible from every other state in $\mathcal{X}$. But, usually this condition is too strong for classical Markovian models for which the transition matrix is sparse.

8 Uniform-Binary decomposition

The aliasing technique, designed by [13], provides an efficient method to build the set of functions $\mathcal{F}$, that simulates the next state following i according to the transition probability $\{p_{i,j}\}_{j\in\mathcal{X}}$. Compared to classical methods [3] such as inverse of probability distribution function (algorithm 3.3), rejection, or composition methods, the complexity of the computation of the next state is in $\mathcal{O}(1)$, and so does not depend on the problem size.

Consider a typical distribution $q = (q_1, \cdots, q_K)$ on K states. The idea is to build a set of K thresholds $\{s_1, \cdots, s_K\}$ $0 \leq s_i \leq 1$, and K couples of states $\{(i_1, i'_1), \cdots, (i_K, i'_K)\}$ with $(i_j, i'_j) \in \mathcal{X}^2$, i'_j is called the alias value of i_j. This construction should verify the following constraints :

$$\forall i \in \{1, \cdots, K\}, \quad q_i = \sum_{j=1}^{K} \left(s_j \mathbb{1}_{i_j=i} + (1 - s_j)\mathbb{1}_{i'_j=i} \right). \tag{5}$$

Such a decomposition is built by a simple algorithm requiring $\mathcal{O}(K)$ steps. The implementation structure is described in [3]. From this structure the simulation runs as follows :

Algorithm 8.1. Aliasing generation

{ The values of s, and couples (i_l, i'_l) are preliminary stored in arrays of size K : S, I and I'.}
$u = Random(0, 1)$;
$v = Random(0, 1)$;
$l = int(u * k)$; { discrimination among K, int means the integer part }
if $v < S[l]$ **then**
  return $I[i]$ { the standard value}
else
  return $I'[i]$ { the alias value}

end if

One should notice that this representation is not unique and, according remarks on the impact of representation on coupling time, we should use heuristics to build a "better" representation. In particular very interesting property of such a construction is that any permutation of two couples $(s_j, (i_j, i'_j))$ and $(s_l, (i_l, i'_l))$ provide another random variable with exactly the same distribution. Moreover, if we replace some threshold s_j by $1 - s_j$ and exchange the values (i_j, i'_j) to (i'_j, i_j), the distribution of the result is also preserved.

Consequently, we have two steps in the computation of the simulation kernel. A first step compute for each state x the corresponding arrays S_i, I_i, and I'_i (cf algorithm 8.1). The second step modify these arrays to guarantee termination in a finite number of steps. To simplify this step, we suppose that there exist some state i_0 such that the transition probability from i_0 to i_0 is strictly positive. This condition is stronger than aperiodicity and is generally verified in practical situations. If not, the matrix $\frac{1}{2}(Id + P)$ exhibits the same stationary distribution as P and could be used instead of P.

Because the Markov chain is irreducible there exist a spanning tree of the state space graph such that a path of positive probability exists in the tree from each state i to i_0. Because each state has an out-degree of 1 in the tree, it is always possible to place the next state of i in the tree (on the path to i_0) in the place $I_i[0]$. Let $\alpha = \min_i \frac{S_i[0]}{d_i}$, with d_i the out-degree of state x in the graph. α is strictly positive and with probability α all the transitions occur on the arrows of the tree. Repeating this transition K times leads to a global coupling in state i_0. The algorithm terminates almost surely. Moreover, if we denote by D the depth of the tree, the coupling time is upper bounded by a geometric distribution with parameter α^D, probability that a burst of D sequential transitions occur on the tree.

The aliasing technique remains valid if some probability are 0 in the distribution q, the corresponding threshold equals 0. Consider that all the alias computations are done on $\{1, \cdots, K\}$ and denote by S and A the threshold and the alias matrices.

Algorithm 8.2. Next states generation (Alias matrices)

$u = Random(0, 1)$;
$k = int(u * K) + 1$;{ choice of the column}
$v = Random(0, 1)$;
for $i = 1$ to n **do**
  if $v < S[i, k]$ **then**
    $NextState(i) = k$ { the standard value}
  else
    $NextState(i) = A[i, k]$ { the alias value}
  end if
end for

For this situation, it appears that the transition matrix have been decomposed in a sum of K stochastic matrices

$$P = \frac{1}{K} (P_1 + \cdots + P_K),$$

where the stochastic matrix P_i have at most two non null elements per row. It corresponds to very simple structures which need further research. The number of matrices in the decomposition could easily be reduced to the maximum out degree of the chain d_{max}, which is useful for sparse matrices. In that case the cardinal of $\mathcal{F}$ is at most $(K+1).d_{max}$.

Moreover using permutations of columns or thresholds could improve the coupling time. The problem of finding the best Uniform-binary RIFS minimizing the mean coupling time seems to be very complex. Some heuristics have been developed but are not yet published.

9 Conclusion

In practical examples, the RIFS representation of a Markov chain is crucial for simulation. Several methods have been implemented in a software PSI [1] in order to test and compare heuristics. All results show that the problem is very difficult and needs further fundamental research.

Experiments on practical performance evaluation problems [12] and [11] shows that Markov chains with up to one million of states could be simulated by this technique. The amount of memory used by the algorithm (alias tables, threshold) is about 2 times the number of positive elements of the transition matrix. Then simulation time is sufficient to estimate parameters on the model typically scala products of a reward vector and the stationary distribution.

When the system is monotone, and that is the case in many practical situations of performance evaluation [11] or in interacting systems of particles [9], the method could be adapted by driving simulation from maximal and minimal states. The coupling time estimation still remains open.

References

[1] A.A. Borovkov and S. Foss. Stochastically recursive sequences and their generalizations. *Siberian Advances in Mathematics*, 2(1), 1992.

[2] A.A. Borovkov and S. Foss. Two ergodicity criteria for stochastically recursive sequences. *Acta Appl. Math.*, 34, 1994.

[3] P. Bratley, B.L. Fox, and L.E. Schrage. *A Guide to Simulation*. Springer-Verlag, 1983.

[4] P. Brémaud. *Markov Chains: Gibbs fields, Monte Carlo Simulation and Queues*. Springer-Verlag, 1999.

[5] P. Diaconis and D. Freedman. Iterated random functions. *SIAM Review*, 41(1):45–76, 1999.

[6] O. Häggström. *Finite markov chains and algorithmic applications*. Cambridge University Press, 2002.

[1] `http://www-id.imag.fr/Logiciels/psi/`

[7] R.M. Loynes. The stability of queues with non independent inter-arrival and service times. *Proc. Cambridge Ph. Soc.*, 58:497–520, 1962.

[8] S.P. Meyn and R.L. Tweedie. *Markov chains and stochastic stability*. Communications and Control Engineering Series. Springer-Verlag, 1993.

[9] J. Propp and D. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1&2):223–252, 1996.

[10] O Stenflo. *Ergodic theorems for Iterated Function Systems controlled by stochastic sequences*. Doctoral thesis n. 14, Umea university, 1998.

[11] J.-M. Vincent. Perfect simulation of queueing networks with blocking and rejection. In *Saint*, pages 268–271, 2005.

[12] J.-M. Vincent and C. Marchand. On the exact simulation of functionals of stationary markov chains. *Linear Algebra and its Applications*, 386:285–310, 2004.

[13] A.J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Software*, 3:253–256, 1974.

Jean-Marc Vincent
Laboratoire ID-IMAG,
MESCAL Inria project,
51, avenue Jean Kuntzmann, F-38330 Montbonnot, France
Jean-Marc.Vincent@imag.fr
http://www-id.imag.fr/Laboratoire/Membres/Vincent_Jean-Marc/

Chapter 3

Extended abstracts

Some numerical analysis problems behind web search

Claude Brezinski Michela Redivo Zaglia

Abstract

An important problem in web search is to classify the pages according to their importance. From the mathematical point of view, Google treats this problem by finding the left principal eigenvector (the PageRank vector) of a certain matrix. Properties of this vector will be given. It could be computed by the power method whose iterates will be characterized. Then several approximations of the PageRank vector, and procedures for accelerating the power method will be discussed.

Introduction

A query to a web search engine often produces a very long list of answers because of the enormous number of pages (over 8 billions in Google's database). To help the surfer, these pages have to be listed starting from the most relevant ones. Google uses several metrics and strategies for solving this *ranking* problem.

The importance of a page is called its *PageRank* and one of the main ingredients of Google's link analysis is the *PageRank algorithm* [3, 6]. A page is considered to be important if many other important pages are pointing to it. So, the importance of a page is determined by the importance of the other pages. This means that the row vector $\mathbf{r}^T$ of all PageRanks is only defined implicitly as the solution of a fixed–point problem, as we will see.

1 The PageRank problem

Let $\deg(i) \geq 1$ be the outdegree (that is, the number of pages it points to) of the page i. Let $P = (p_{ij})$ be the matrix which describes the transition from the page i to the page $j \neq i$ with $p_{ij} = 1/\deg(i)$, and $p_{ii} = 0$.

The PageRank vector $\mathbf{r}^T$ satisfies $\mathbf{r}^T = \mathbf{r}^T P$, that is, $\mathbf{r} = P^T \mathbf{r}$, and it can be computed recursively by the standard power method

$$\mathbf{r}^{(n+1)} = P^T \mathbf{r}^{(n)}, \quad n = 0, 1, \ldots,$$

assuming that all the eigenvectors of P^T are present in the spectral decomposition of $\mathbf{r}^{(0)}$. Unfortunately, this iterative procedure has convergence problems.

For avoiding these drawbacks, the original PageRank algorithm was modified. First, since some pages have no outlink, P is not stochastic. So, P is replaced by another matrix $\widetilde{P}$. Let $\mathbf{w} = (w_1, \ldots, w_p)^T \in \mathbb{R}^p$ be a probability vector, that is such that $\mathbf{w} \geq 0$ and

$\mathbf{e}^T\mathbf{w} = 1$ with $\mathbf{e} = (1, \ldots, 1)^T$, and p the total number of pages. Let $\mathbf{d} = (d_i) \in \mathbb{R}^p$ be the vector with $d_i = 1$ if $\deg(i) = 0$, and 0 otherwise. We set

$$\widetilde{P} = P + \mathbf{d}\mathbf{w}^T.$$

The effect of the additional matrix $\mathbf{d}\mathbf{w}^T$ is to modify the probabilities so that a surfer visiting a page without outlinks jumps to another page with the probability distribution defined by $\mathbf{w}$. Thus, $\widetilde{P}$ is stochastic, and has 1 as a dominant eigenvalue with $\mathbf{e}$ as its corresponding right eigenvector.

Another problem arises since $\widetilde{P}$ is reducible. In that case, $\widetilde{P}$ can have several eigenvalues on the unit circle, thus causing convergence problems to the power method. Moreover, $\widetilde{P}$ can have several left eigenvectors corresponding to its dominant eigenvalue 1.

Then, $\widetilde{P}$ itself is finally replaced by the matrix

$$P_c = c\widetilde{P} + (1 - c)E, \qquad E = \mathbf{e}\mathbf{v}^T,$$

with $c \in [0, 1]$, and $\mathbf{v}$ a probability vector. It corresponds to adding to all pages a new set of outgoing transitions with small probabilities. The probability distribution given by the vector $\mathbf{v}$ can differ from a uniformly distributed vector, and the resultant PageRank can be biased to give preference to certain kinds of pages. The matrix P_c is stochastic and irreducible since $\mathbf{v}$ is a positive vector. P_c has an eigenvalue equal to 1 with $\mathbf{e}$ as its corresponding right eigenvector. Indeed

$$P_c\mathbf{e} = c\widetilde{P}\mathbf{e} + (1 - c)\mathbf{e}\mathbf{v}^T\mathbf{e} = c\mathbf{e} + (1 - c)\mathbf{e} = \mathbf{e}.$$

The power iterations for the matrix P_c^T now converge to a unique vector $\mathbf{r}_c$ (obviously, depending on c) which is chosen as the PageRank vector.

2 The power method

Thus, we are faced to the following mathematical problem. For consistency to prior works, we set $A_c = P_c^T$.

The $p \times p$ matrix A_c has eigenvalues $|\lambda_p| \leq \cdots \leq |\lambda_2| < \lambda_1 = 1$, and we have to compute $\mathbf{r}_c$, its unique right eigenvector corresponding to the eigenvalue $\lambda_1 = 1$. For that purpose, we use the power method which consists in the iterations

$$\mathbf{r}_c^{(n+1)} = A_c\mathbf{r}_c^{(n)}, \qquad n = 0, 1, \ldots$$

with $\mathbf{r}_c^{(0)}$ given.

The sequence $(\mathbf{r}_c^{(n)})$ always converges to $\mathbf{r}_c$ but, if $c \simeq 1$, the convergence is slow since the power method converges as c^n. So, a balance has to be found between a small value of c, which insures a fast convergence of $(\mathbf{r}_c^{(n)})$, but to a vector $\mathbf{r}_c$ which is not close to the real PageRank vector $\widetilde{\mathbf{r}} = \lim_{c\to 1} \mathbf{r}_c$, and a value of c close to 1, which leads to a better approximation $\mathbf{r}_c$ of $\widetilde{\mathbf{r}}$, but with a slow convergence. Google usually chooses $c = 0.85$, which insures a good rate of convergence.

3 Approximation and acceleration

Since computing a PageRank vector can take several days, convergence acceleration or approximations of the PageRank vector are essential, in particular, for providing continuous updates to ranking. Moreover, some recent approaches require the computation of several PageRank vectors corresponding to different personalization vectors. Recently, several methods for accelerating the computation of the PageRank vector by the power method were proposed [5, 4]. The aim of this work is to give a theoretical justification to the methods of [5], and to put them on a firm theoretical basis. We will interpret them in a different way, and simplify, unify, and generalize them. In particular, we will explain their connection with the method of moments of Vorobyev. Other possible acceleration procedures will also be discussed.

Another problem related to PageRank computations is that, as c approaches 1, the matrix A_c becomes more and more ill conditioned since its condition number behaves as $(1-c)^{-1}$, the conditioning of the eigenproblem becomes poor, and $\mathbf{r}_c$ cannot be computed accurately. So, $\mathbf{r}_c$ can be computed for several values of c far away from 1 by any procedure, and then these vectors can be extrapolated at the point $c = 1$ (or at any other point). In order for an extrapolation procedure to work well, the extrapolating function has to mimic as closely as possible the behaviour of $\mathbf{r}_c$ with respect to the parameter c.

Since P_c is stochastic and irreducible, $\mathbf{r}_c$ is the unique right eigenvector of $A_c = P_c^T$ corresponding to the eigenvalue 1, that is, $A_c\mathbf{r}_c = \mathbf{r}_c$. We have $\mathbf{r}_c \geq 0$, and we normalize it such that it is a probability vector, that is $\mathbf{e}^T\mathbf{r}_c = 1$.

So, we will study the properties of this vector, and, in particular, we will give implicit and explicit expressions for it. Then, we will discuss its computation by the power method. The iterates given by the power method are, in fact, the partial sum of a power series with vector coefficients [1]. This discussion will lead us to various procedures for accelerating the convergence of the power method, and to processes for the approximation of the PageRank vector. In particular, the iterates of the power method, which are in fact the partial sums of a vector formal power series, will be used for constructing Padé style approximations of $\mathbf{r}_c$. The convergence of the power method itself will be accelerated by using various vector sequences transformations, in particular, the ε-algorithms and Aitken's Δ^2 process. The acceleration processes proposed in [5] will be put on a firm theoretical basis and explained in the framework of the method of moments. They will also be generalized.

All these results are explained in details in [2].

References

[1] P. Boldi, M. Santini, S. Vigna, PageRank as a function of the damping factor, Poster Proceedings of the 14th International World Wide Web Conference, May 10-14, 2005, Chiba, Japan.

[2] C. Brezinski, M. Redivo–Zaglia, The PageRank vector: properties, computation, approximation, and acceleration, submitted.

[3] S. Brin, L. Page, The anatomy of a large–scale hypertextual web search engine, Comput. Networks ISDN Syst., 30 (1998) 107–117.

[4] S. Kamvar, T. Haveliwala, G. Golub, Adaptive methods for the computation of PageRank, Linear Algebra Appl., 386 (2004) 51–65.

[5] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, G.H. Golub, Extrapolations methods for accelerating PageRank computations, in *Proceedings of the Twelfth International World Wide Web Conference*, ACM Press, 2003, pp. 261-270.

[6] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: bringing order to the Web, Stanford University Technical Report, 1999, http://dbpubs.stanford.edu/pub/1999-66

Claude Brezinski
Laboratoire Paul Painlevé, UMR CNRS 8524
UFR de Mathématiques Pures et Appliquées
Université des Sciences et Technologies de Lille
59655 - Villeneuve d'Ascq cedex
France
claude.brezinski@univ-lille1.fr

Michela Redivo Zaglia
Dipartimento di Matematica Pura ed Applicata
Università degli Studi di Padova
Via G.B. Belzoni, 7
35131 - Padova
Italy
Michela.RedivoZaglia@unipd.it

Computational complexity of numerical solution of polynomial systems

Robert M. Corless Silvana Ilie Greg Reid

The problem of finding all isolated roots of polynomial systems is of major interest. Our aim is to compute the cost of finding all isolated roots of a polynomial system. By using the method presented below, each isolated root can be computed independently, and therefore sequential computation of all isolated roots is possible (and the cost of computing them is the sum of each individual cost of computing a root). This can be achieved provided that all the isolated roots have multiplicity one.

The case of higher multiplicity for an isolated root is not treated here. We mention that there exist tools to detect multiplicity and methods to remove it (called "deflation of multiplicity") e.g., differentiation of the system, see Lecerf [9], Leykin, Verschelde & Zhao [10], Dayton & Zeng [5].

We focus on finding a low (polynomial) cost algorithm of approximating one isolated regular root of a polynomial system depending on one parameter, namely the number of digits of accuracy requested for the residual. The study of the complexity of approximating all the isolated roots of a generic system of polynomials with respect to other parameters was considered by, e.g., Shub & Smale [12]. For recent interesting work on location of zeros for analytic functions we refer the reader to Giusti, Lecerf, Salvy & Yakoubsohn [6].

We use the homotopy continuation method for finding all isolated roots of a polynomial system. The method consists in finding the numerical solution by continuously deforming the target system from a starting system with known solutions. A significant accomplishment of homotopy theory has been to give specific forms of the (new) start systems which guarantee that all isolated roots of the target system can be recovered by tracking the paths starting from the known solutions (see, e.g., [1, 13, 14]).

The path tracking problem may be obtained by applying Pryce's method for differential algebraic equation (DAE) solving [11] directly to the case of polynomial systems or indirectly, after transforming the problem to an index-1 DAE, to a problem to which the method applies. We shall choose the second approach and use a new result [8] which showed that numerical solutions of initial value problems (IVP) for index-1 DAE can be computed in polynomial time. This result is valid for problems involving piecewise analytic functions. It extends a recent result by Corless [3] showing that there exist algorithms of polynomial cost in the number of digits of accuracy to solve IVP for ODE.

The particular choice of the homotopy guarantees with probability one that the system Jacobian is non-singular along the path. Therefore, we may assume that the path stays away from the locus corresponding to singular Jacobian and thus Pryce's method applies.

Consequently, by assuming that such a curve exists and the Jacobian does not become singular along the path, we use the intrinsic smoothness of the problem and we find an algorithm of cost polynomial in the number of bits of accuracy for numerically approximating one isolated root of the given polynomial system.

Semi-explicit index-1 DAE Consider the following semi-explicit index-1 DAE:

$$y'(t) = f(y(t), z(t)) \quad (1)$$
$$0 = g(y(t), z(t)) \quad (2)$$

for $t \in I = (a, b)$, $y : I \to \mathbb{R}^k$, $z : I \to \mathbb{R}^m$, $f : D \subset \mathbb{R}^{m+k} \to \mathbb{R}^k$ and $g : D \subset \mathbb{R}^{m+k} \to \mathbb{R}^m$, where D is an opened subset of $\mathbb{R}^{m+k}$. We assume that the solution path lies in D and that g_z is invertible along the solution path. The initial conditions are assumed consistent with the constraints. We assume that the solution exists, is unique and analytic. We also assume that f and g are computable with automatic differentiation.

Under some natural regularity conditions it can be shown [8] that:

Theorem 1. *For sufficiently small ε, the minimal cost of obtaining the solution with residual error ε of the initial value problem for (1)-(2) using Pryce's method is polynomial in the number of digits of accuracy. The minimum is reached on the equidistributing mesh.*

Theorem 1 can be shown to hold also for piecewise analytic functions provided that the breaking points are asymptotically, as the tolerance becomes small, $\mathcal{O}(1)$ appart. Furthermore, the theorem can be extended to complex–valued functions and variables.

Homotopy continuation Consider a system of polynomials (the 'target system') which we want to solve

$$p = (p_1, \ldots, p_m)^T = 0, \quad (3)$$

where $p : \mathbb{C}^m \to \mathbb{C}^m$ and $x = (x_1, \ldots, x_m) \in \mathbb{C}^m$.

Assume there exists a polynomial system (a 'starting system') whose solutions are all isolated and known, $q(x) = 0$. The following homotopy is considered

$$H(t, x) = (1 - t)q(x) + tp(x) \ . \quad (4)$$

The aim is to solve the problem

$$H(t, x(t)) = 0, t \in [0, 1]. \quad (5)$$

The starting point of the problem (5) is chosen to be a known solution $x_0 \in \mathbb{C}^m$ of the starting system. At the end of the solution path, the roots of the target system are obtained.

Once the system p is given, one can construct a polynomial system q, such that the homotopy (4) is a "good homotopy" (see also Verschelde [14], Definition 1.2.4).

Definition 2. A good homotopy $H(t, x)$ should satisfy the following properties:

1. (triviality) The solutions at $t = 0$ are known.
2. (smoothness) The solution set of $H(t, x(t)) = 0$ for all $t \in [0, 1]$ consists of a finite number of smooth paths, each parametrized by t.
3. (accessibility) Every isolated solution of $H(1, x) = 0$ is reached by some path originating at a solution of $H(0, x) = 0$. If r is the multiplicity of an isolated solution x^*, then there are exactly r path converging to x^*.

For arbitrary target systems $p(x)$, methods for constructing good start systems are known (see, e.g., [13]).

The basic ideas behind showing that a homotopy is a good homotopy go back to the breakthrough paper [2], which shows that the number of roots is bounded by the mixed volume (which may be much lower than the Bèzout number). Since then, there has been much progress in finding efficient choices for the start systems ([13] and references therein) based on mixed volumes. Such choices not only find all isolated solutions, but also minimize and, in many cases, eliminate diverging paths.

The issue of optimal starting polynomial system selections depending on the structure of the given polynomial system p (such as density, sparsity, etc.) for constructing good homotopies is not discussed here.

A start system q for which the linear homotopy (4) is a good homotopy is given by, eg., Theorem 4.1.13 in [14]. For other examples of classes of start systems and good homotopies see [13] and its references. We assume below that q is such that (4) is a good homotopy.

Numerical solution The numerical solution of (5) is obtained as values at the points of the mesh, which can be interpolated so that a continuous extension is obtained, $x_j(t)$, with $j = 1, \ldots, m$. Consequently, we can define the residual in the computed solution as

$$\delta_j(t) = H_j(t, x(t)), \text{ for all } j = 1, \ldots, m \text{ and all } t \in [0, 1]. \tag{6}$$

Given a small tolerance ε, we require that along the path the residual satisfies the tolerance $\|\delta(t)\| \leq \varepsilon$, for all $t \in [0, 1]$. Since the Jacobian H_x is non-singular along solution path, then, with a smoothness and a compactness argument, there exist $M > 0$ and a neighborhood of the exact solution path such that $\|H_x^{-1}(t, y)\| < M$ for all (t, y) in that neighborhood. Therefore, for small enough tolerance ε,

$$\|x(t) - x^e(t)\| \leq M\|H(t, x(t))\| \leq M\varepsilon$$

for all $t \in [0, 1]$ where $x^e(t)$ is the exact solution of (5).

For solving numerically (5), Pryce's structural analysis [11] is used. Pryce's method, which is based on Taylor series, can be extended for complex-valued unknown variables and unknown functions.

The problem can be written as a semi-explicit autonomous index-1 DAE

$$\left\{ \begin{array}{rcl} t' & = & 1 \\ 0 & = & H(t, x) \ . \end{array} \right. \tag{7}$$

Note that the system Jacobian in Pryce's method (i.e. $\mathcal{J} = \begin{bmatrix} 1 & 0 \\ 0 & H_x \end{bmatrix}$) of (7) is non-singular provided that the Jacobian H_x is non-singular. A crucial aspect of the theoretical justification of "good" homotopy (see [2, 13]) is to show that all the isolated roots are at the end of smooth homotopy path with H_x non-singular along the path ($0 \leq t < 1$). The additional requirement that the root is of multiplicity 1 yields that H_x is non-singular for $0 \leq t \leq 1$. The non-singularity of $\mathcal{J}$ implies that Pryce's structural analysis applies (see [7]).

As a consequence, applying Theorem 1 on (7) gives the following (see also [7]):

Theorem 3. *For sufficiently small ε, the minimal cost of numerically solving the problem (5) with error ε using an algorithm based on Pryce's structural analysis is polynomial in the number of bits of accuracy.*

Remark 4. We note that variable order of the method is necessary to guarantee polynomial cost (it is chosen to be $(1/2)\ln(1/\varepsilon)$). Also, it can be shown that the number of steps in the mesh depends weakly on the number of digits of accuracy, through some norm of local error coefficients.

Remark 5. The polynomial cost method used in this paper applies in its fully generality to real analytic and complex analytic differential algebraic equations. Consequently, a natural question is to consider analytic system of equations via homotopy methods. Provided a "good homotopy" is available, the polynomial computational cost results still apply. However, theoretically validated "good homotopies" for general analytic systems are not known and results have to be on a case by case basis (also see [6] for validated analytic solving in the univariate case).

Conclusions We have shown that there exist algorithms of finding all isolated roots of a polynomial system which are polynomial in the number of digits of accuracy requested. The problem of roots finding is solved with homotopy continuation methods. Our result relies on a new computational complexity result for numerically solving IVP for index-1 DAE.

A more detailed analysis of other parameters affecting the computational cost (e.g. sparsity and degrees of the input polynomials, number of unknowns) is an important problem which should be investigated in future work. It is also interesting to treat the case of higher multiplicity roots by employing deflation methods. The computational cost of the deflation should be estimated (see [9] for comments on the exact case).

References

[1] E.L. Allgower and K. Georg, Numerical path following, in: P.G. Ciarlet and J.L. Lions, eds., *Techniques of Scientific Computing (Part 2)*, Vol. 5 of *Handbook of Numerical Analysis*, North-Holland, 1997, 3 – 203.

[2] D.N. Bernstein, The number of roots of a system of equations, *Functional Anal. Appl.* **9**(3) (1975) 183 – 185.

[3] R.M. Corless, A new view of the computational complexity of IVP for ODE, *Numerical Algorithms* **31** (2002) 115 – 124.

[4] R.M. Corless, M.W. Giesbrecht, M. van Hoeij, I.S. Kotsireas, and S.M. Watt, Towards factoring bivariate approximate polynomials, *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, ACM, New York, 2002, 37 – 45.

[5] B.H. Dayton, Z. Zeng, Computing the multiplicity structure in solving polynomial systems, *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation*, ACM, New York, 2005, 116 –123.

[6] M. Giusti, G. Lecerf, B. Salvy, and J.-C. Yakoubsohn, On location and approximation of clusters of zeros of analytic functions. Found. Comput. Math. 5 (2005), no. 3, 257–311.

[7] S. Ilie, Computational complexity of numerical solutions of IVP for DAE, PhD. Thesis, University of Western Ontario, Nov. 2005.

[8] S. Ilie, R.M. Corless, and G. Reid, Numerical solutions of index-1 differential algebraic equations can be computed in polynomial time, *Numerical Algorithms* **41**(2) (2006) 161 – 171.

[9] G. Lecerf, Quadratic Newton iteration for systems with multiplicity, *Found. Comput. Math.*, **2**, (2002) 247 –293.

[10] A. Leykin, J. Verschelde, A. Zhao, Newton's method with deflation for isolated singularities of polynomial systems, *submitted*.

[11] J.D. Pryce, A simple structural analysis method for DAEs, *BIT* **41**(2) (2001) 364 – 394.

[12] M. Shub and S. Smale, Complexity of Bézout's theorem V: polynomial time, *Theoretical Computer Science* **133**(1) (1994) 141 – 164.

[13] A.J. Sommese and C.W. Wampler, *The numerical solution of systems of polynomials arising in engineering and science*, World Scientific Press, Singapore, 2005.

[14] J. Verschelde, Homotopy continuation methods for solving polynomial systems, Ph.D. thesis, Katholieke Universiteit Leuven, 1996.

Robert M. Corless
Ontario Research Centre for Computer Algebra
University of Western Ontario
E-mail: rcorless@uwo.ca
http://www.apmaths.uwo.ca/~rcorless/

Silvana Ilie
Ontario Research Centre for Computer Algebra
University of Western Ontario
E-mail: silvana@uwo.ca
http://publish.uwo.ca/~silvana/

Greg Reid
Ontario Research Centre for Computer Algebra
University of Western Ontario
E-mail: reid@uwo.ca
http://www.orcca.on.ca/~reid/

On Fredholm property of elliptic PDEs

Katya Krupchyk Jukka Tuomela

Abstract

Let us consider elliptic boundary value problems in some domain. When studying the well-posedness of such problems it is convenient to relax the requirement of existence and uniqueness, and allow a finite dimensional kernel and cokernel to our operators. This kind of operators are called Fredholm operators. In case of square systems (as many equations as unknowns) such problems are Fredholm in appropriate Sobolev spaces if and only if the boundary conditions satisfy the Shapiro–Lopatinskij condition. This result is valid also for elliptic systems in the generalised sense of Douglis and Nirenberg. In case of overdetermined systems one needs to construct the compatibility complex of the boundary value problem operator, and now one asks under which condition the resulting complex is Fredholm, i.e. when the cohomologies of the complex are finite dimensional.

To study the overdetermined systems there is an analog of the Shapiro-Lopatinskij condition which gives sufficient conditions for the compatibility complex to be Fredholm. It was thought that this generalized Shapiro-Lopatinskij condition is also necessary. However, we present an example which shows that this is not true. Moreover, we propose a constructive algebraic criterion for checking the (generalised) Shapiro-Lopatinskij condition and show how to compute the compatibility complex for operators with constant coefficients.

We study the connection between ellipticity and Fredholm property of the boundary value problem operators on manifolds with boundary. Ellipticity is classically defined by using the principal symbol of the operator. We will only consider linear problems which can be written (in a suitable coordinate system) in the following general form

$$Ay = \sum_{|\mu| \leq q} a_\mu(x) \partial^\mu y = f$$

Here a_μ are matrices of size $k \times m$ (with $k \geq m$) which a priori may depend on the point x of the domain. However, from now on we will suppose that various properties which we consider do not depend on x. The principal symbol of this system is then

$$\sigma \mathsf{A}(x, \xi) = \sum_{|\mu| = q} a_\mu(x) \xi^\mu.$$

Here also we will suppress the dependence on x. Now the operator is elliptic, if the principal symbol is injective for all $\xi \neq 0$. In order to have a well defined problem we need also a

differential operator on the boundary which gives the boundary conditions. This operator is denoted by B.

We are interested in determining when the boundary value problem defined by A and B is Fredholm, i.e. has a finite dimensional kernel and conkernel. In the square case the answer is well-known: the operators should satsify the Shapiro–Lopatinskij condition. This condition can be formulated algebraically in terms of symbols of both A and B.

Douglis and Nirenberg [3] generalized the notion of ellipticity of operators. They introduced some weights for equations and unknowns so that their symbol contains information also about derivatives which are not of maximal order. It turns out the theory of Shapiro–Lopatinskij conditions can be generalised also to this case.

The Fredholm property of elliptic boundary value problems is well-known [1] in the square case, i.e. we have as many equations as unknowns. In the overdetermined case the situation is less clear. First in the overdetermined case the cokernel is typically infinite dimensional, so the standard Fredholm property fails. However, when we construct the compatibility complex to our boundary value problem operator we may inquire if the resulting complex has finite dimensional cohomologies. If this is the case we say that the complex is Fredholm and also that the original problem is Fredholm.

Now to analyse overdetermined PDEs in general it is important to check if the system is involutive, and if not then transform it to the involutive form. The technical definition of the involutive form is quite complicated (see [8], [9] and [10] and for the actual definition). The geometrical definition is based on representing the PDE system as subbundle of some appropriate jet space. However, essentially the involutivity means that one has to find all integrability conditions (or differential consequences) of the given system up to some order. Under some appropriate hypothesis one can show that this can be done in a constructive way; this is sometimes called the Cartan–Kuranishi completion algorithm.

The transformation to the involutive form usually requires the use of symbolic computation. In practice to complete a system to the involutive form one may use DETools package [2] in computer algebra system MuPAD [5].

Now when we complete our system to involutive form we see that we no longer need the weights of Douglis and Nirenberg: in [6] we showed that any system that is elliptic with respect to the generalised definition becomes elliptic in the standard sense when completed to the involutive form. So the apparent generality of ellipticity is just the result of restricting the attention to square systems. Moreover, we gave examples of operators that are not elliptic (even in the generalised sense) but whose involutive form are elliptic. Hence when determining ellipticity of the operator one should consider its involutive form and check the classical ellipticity.

It turns out that for technical reasons it is convenient to further transform the involutive system to a normalised system. Roughly speaking, an operator is normalised if it is a first order involutive operator and there are no (explicit or implicit) algebraic (i.e., non-differential) relations between dependent variables. A boundary value problem operator is normalised if the system is normalised and the boundary conditions contain only differentiation in directions tangent to the boundary. Evidently this transformation is constructive so there is no loss of generality in assuming the our system is normalised.

Recall that we are interested in studying if the cohomologies of compatibility complexes are finite dimensional or not. It is an important fact that when we first compute the involutive form and then the normalised form, the dimensions of the cohomology spaces remain invariant, or in the language of homology theory, the resulting complexes are homotopically equivalent.

Then we have to construct the compatibility complex. Let us first consider the operator A itself. For simplicity we will only discuss the case when the operators (and later boundary operators) have constant coefficients. In this case we can compute the compatibility complex by simply computing the free resolution of the module generated by the rows of A. Incidentally this shows that the length of the compatibility complex is at most the number of independent variables. However, to study boundary value problems we need to compute the compatibility operators involving the boundary operators. This is not as straightforward as the simple free resolution, and in fact here we need the notion of normalised operator to perform this task. Anyway the problem can be formulated again with modules, and choosing suitable module orderings we can compute the necessary information by Gröbner basis techniques.

Having all the above information available one can generalize the Shapiro-Lopatinskij condition to the overdetermined case. It turns out that to check this condition we can propose a similar criterion as in the standard case. The most difficult part in using the criterion is that we need to factorise the characteristic polynomial of the system (in the standard square case this is the determinant of the symbol). Note that this difficulty is already present in the square case. In our criterion we need to check ranks of some matrices which are obtained from symbols of A, B and relevant compatibility operators. Moreover, in case of two independent variables we propose also a computer code (written in computer algebra system Singular) for checking our criterion. Note that this case is already important in PDEs theory.

Now it is known that (under some appropriate technical hypothesis) if a boundary value problem operator satisfies the generalised Shapiro-Lopatinskij condition then its compatibility complex is Fredholm [4] (for example in appropriate Sobolev spaces). It was thought that this generalized Shapiro-Lopatinskij condition is also necessary. However, we recently found that this is not true and in fact, the familiar (stationary) Stokes problem provides us with a counterexample. Note that this is not a rare phenomenon: now that we understand what goes wrong with the Stokes system we can easily construct other examples. By the same techniques we can also show that if a nonelliptic system has an elliptic involutive form then the original problem is in fact Fredholm.

This work is a continuation to our paper [7].

References

[1] M.S. Agranovich, *Elliptic Boundary Problems*, Partial Differential Equations IX, Springer-Verlag, (ed. M.S. Agranovich and Yu.V. Egorov and M.A. Shubin), 1997, Encyclopaedia of Mathematical Sciences 79, Berlin/Heidelberg, pp. 1–144.

[2] J. BELANGER AND M. HAUSDORF AND W. SEILER, *A MuPAD Library for Differential Equations*, Computer Algebra in Scientific Computing — CASC 2001, Springer-Verlag, Berlin/Heidelberg, (eds. V.G. Ghanza and E.W. Mayr and E.V. Vorozhtsov), (2001), 25–42.

[3] A. Douglis and L. Nirenberg, *Interior Estimates for Elliptic Systems of Partial Differential Equations*, Comm. Pure Appl. Math., **8** (1955), 503–538.

[4] P.I. Dudnikov and S.N. Samborski, *Linear Overdetermined Systems of Partial Differential Equations. Initial and Initial-Boundary Value Problems*, Partial Differential Equations VIII, (ed. M.A. Shubin), Springer-Verlag, Berlin/Heidelberg, 1996, Encyclopaedia of Mathematical Sciences 65, pp. 1–86

[5] J. GERHARD AND W. OEVEL AND F. POSTEL AND S. WEHMEIER, MuPAD *tutorial*, Springer, (2000), `http://www.mupad.de/`.

[6] K. Krupchyk, W. Seiler and J. Tuomela, *Overdetermined elliptic PDEs*, to appear in J. Found. Comp. Math.

[7] K. Krupchyk and J. Tuomela, *Shapiro-Lopatinskij condition for elliptic boundary value problems*, submitted to LMS J. Comput. Math.

[8] J. F. POMMARET, *Systems of Partial Differential Equations and Lie Pseudogroups*, Mathematics and its applications, **14**, Gordon and Breach Science Publishers, (1978).

[9] N. TARKHANOV, *Complexes of Differential Operators*, Kluwer Academic Publishers, Dordrecht, NL, 1995.

[10] D. SPENCER, *Overdetermined systems of linear partial differential equations*, Bull. Am. Math. Soc, **75**, (1969), 179–239.

Katya Krupchyk
Department of Mathematics
University of Joensuu, Finland
katya.krupchyk@joensuu.fi
http://www.joensuu.fi/mathematics/department/personnel/krupchyk.htm

Jukka Tuomela
Department of Mathematics
University of Joensuu
jukka.tuomela@joensuu.fi
http://www.joensuu.fi/mathematics/department/personnel/tuomela.htm

Component-free vector algebra in Aldor

Songxin Liang David J. Jeffrey Stephen M. Watt

Abstract

An implementation of a component-free symbolic vector algebra in Aldor is presented. This package provides two powerful functions: simplification of vector expressions and the proof of vector identities. The implementation benefits greatly from Aldor's strong typing, which allows several simplification problems that have defeated previous implementations to be solved.

1 Introduction

Vector algebra and vector calculus have many applications throughout science and mathematics. Vector analysis often simplifies the derivation of mathematical theorems and the statements of physical laws, while vector notation can often clearly convey geometric or physical interpretations that greatly facilitate understanding. Physicists and engineers prefer to formulate their equations of motion using abstract vectors rather than components. Thus they prefer to write the velocity of a rotating body as $\omega \wedge \mathbf{r}$, rather than in some component form as

$$[\omega_2 z - \omega_3 y, \omega_3 x - \omega_1 z, \omega_1 y - \omega_2 x] \ .$$

Many computer algebra systems provide vector operations, and some have add-on vector analysis packages, for example, the VectorAnalysis package for Mathematica [4]. See also [2]. However, these packages do not perform component-independent operations. This means that before one can do any vector operations, one must set the components for all vectors involved. Component-independent systems are difficult and challenging, because vector algebra has a strange and intriguing structure [3]. Denoting the vector and scalar products of a, b by $a \wedge b$ and $a \cdot b$, we see:

- neither operation is associative. If p, q, r are vectors, then $p \wedge (q \wedge r) \neq (p \wedge q) \wedge r$, whereas $p \cdot (q \cdot r)$ and $(p \cdot q) \cdot r$ are invalid.
- neither operation has a multiplicative unit. There does not exist a fixed vector u such that for any vector p, $u \wedge p = p$ or $p \wedge u = p$ or $u \cdot p = p$ or $p \cdot u = p$.
- both admit zero divisors. For any vector p, $p \wedge p = 0$; if q is a vector perpendicular to p, then $p \cdot q = 0$.
- the two operations are connected by the strange side relation $p \wedge (q \wedge r) = (p \cdot r)q - (p \cdot q)r$

Stoutemyer [3] implemented component-free vector operations. However, Stoutemyer's package left some simplification problems unsolved. When he tried to simplify the vector expression

$$(a \wedge b) \wedge (b \wedge c) \cdot (c \wedge a) - (a \cdot (b \wedge c))^2$$

which should simplify to zero, he only got

$$a \cdot (a \cdot b \wedge c.b) \wedge c - (a \cdot b \wedge c)^2 \ .$$

The reason was that although the scalar factor $a \cdot b \wedge c$ could be factored out, revealing the expression to be zero, the built-in scalar-factoring-out mechanism could not recognize that $a \cdot b \wedge c$ is a scalar despite its vector components.

Because of space limitations, this description of the program is necessarily brief, and omits most details. However, the source code for the program will be made available on the Aldor web site, from where it may be downloaded and inspected.

2 Mathematical strategy

In this section we describe the mathematical strategy of the package, while the next section describes the Aldor implementation. Simplification is achieved by defining a canonical form for a vector expression and transforming all input expressions into this form.

2.1 Expression representation

A vector expression is a sum of terms, with each term being a product of scalar and vector. The scalar is further divided into scalars from the coefficient field and scalars formed because of a scalar product of vectors. A vector expression is represented as: `Rep==List Term,` and

```
Term==Record (coe:R, sca:  List List String, vec:List String),
```

where `coe, sca, vec` are respectively the coefficient, scalar part and vector part of a term. For example, the term $-2(a.b)(a \wedge b \cdot c)(c \wedge d)$ is represented as

```
[-2, [["a","b"], ["a","b","c"]], ["c","d"]].
```

Note that the vector triple product $(a \wedge b) \cdot c$ is an important scalar that is represented by the three-element bracket. Because of the equality $(a \wedge b) \cdot c = a \cdot (b \wedge c)$, the product can be represented as the triple `["a","b","c"]`.

In addition to the basic data structure, there is a set of ordering rules to two occurrences of the same term will be represented by the same list.

2.2 Transformation rules

The transformation rules used to reduce expressions to canonical form are taken from standard textbooks. In the current version of the program, the basic rules are not all independent of each other. For example, the program applies separately the rule

$$a \wedge a = 0 \ ,$$

even though it can be deduced from the rule

$$a \wedge b = -b \wedge a \ ,$$

because setting $b = a$ gives $a \wedge a = -a \wedge a$. Note that in physics books, the rule is proved by using the fact that the angle between parallel vectors is 0.

3 Implementation in Aldor

Aldor is a strongly typed, imperative programming language with a two-level object model of categories and domains [1]. Here we give an overview of a few of the top level constructs. We first define a vector space category as follows:

```
define VectorSpcCategory(R:Join(ArithmeticType, ExpressionType), n:MI==3):
Category==with
{
  *: (R,%)->%;
  *: (%,R)->%;
  +: (%,%)->%;
  -: (%,%)->%;
  -:     %->%;
  =: (%,%)->Boolean;
  default
  {
    import from R;
    (x:%)-(y:%):%==x+(-1)*y;
    (x:%)*(r:R):%==r*x;
    -(x:%):%==(-1)*x;
  }
}.
```

Here, n is the dimension of the space. Based on VectorSpcCategory, we define the vector algebra category VectorAlgCategory as follows.

```
define VectorAlgCategory(R:Join(ArithmeticType, ExpressionType)):
Category== VectorSpcCategory(R) with
{
  vector:      Symbol->%;
  scalarZero:  ()->%;
  vectorZero:  ()->%;
  realVector?: %->Boolean;
  simplify:    (%,UseAdvancedRules:Boolean==false)->%;
  identity?:   (%,%)->Boolean;
  s3p:   (%,%,%)->%;
```

```
  *:     (%,%)->%;
  apply: (%,%)->%;
  ^:     (%,%)->%;
  <<:    (TextWriter,%)->TextWriter;
  default
  {
    s3p(x:%,y:%,z:%):%==apply(x^y,z);
  }
}.
```

With these categories, we can implement a vector algebra domain `VectorAlg`.

4 An Example

Stoutemyer's unsolved problem from section 1 is solved as follows.

$$\begin{aligned}
((a \wedge b) \wedge (b \wedge c)) \cdot (c \wedge a) - (a \cdot (b \wedge c)) * (a \cdot (b \wedge c)) &\Rightarrow \\
((a \wedge b \cdot c) * b - (a \wedge b \cdot b) * c) \cdot (c \wedge a) - (a \cdot (b \wedge c)) * (a \cdot (b \wedge c)) &\Rightarrow \\
((a \wedge b \cdot c) * b - 0) \cdot (c \wedge a) - (a \cdot (b \wedge c)) * (a \cdot (b \wedge c)) &\Rightarrow \\
(a \wedge b \cdot c) * (b \cdot (c \wedge a)) - (a \cdot (b \wedge c)) * (a \cdot (b \wedge c)) &\Rightarrow \\
(a \wedge b \cdot c) * (b \wedge c) \cdot a - (a \cdot (b \wedge c)) * (a \cdot (b \wedge c)) &\Rightarrow \\
(a \wedge b \cdot c) * (a \cdot (b \wedge c)) - (a \cdot (b \wedge c)) * (a \cdot (b \wedge c)) &\Rightarrow \\
(a \cdot (b \wedge c) * (a \cdot (b \wedge c)) - (a \cdot (b \wedge c)) * (a \cdot (b \wedge c)) &\Rightarrow 0
\end{aligned}$$

References

[1] Aldor Compiler User Guide, `http://www.aldor.org`

[2] Fiedler, B., 1997. Vectan 1.1. Manual Math. Inst., Univ. Leipzig, 1-22.

[3] Stoutemyer, D. R., 1979. Symbolic computer vector analysis. Computers & Mathematics with Applications, v 5, n 1, 1979, p 1-9.

[4] Wolfram, S., 1996. The Mathematica Book, 3rd ed.. Wolfram Media/Cambridge University Press.

S. Liang, D.J. Jeffrey, S.M. Watt
Ontario Research Centre for Computer Algebra,
The University of Western Ontario, London, Ontario, Canada N6A 5B7
djeffrey@uwo.ca
`http://www.apmaths.uwo.ca/~djeffrey/`

Primary decomposition of zero-dimensional ideals: Putting Monico's algorithm into practice

Marc Moreno Maza Éric Schost Wenqin Zhou

Abstract

Monico published in [*Journal of Symbolic Computation*, 34(5):451–459, 2002] an algorithm to compute the primary decomposition of a zero-dimensional ideal, that mostly relies on a characteristic polynomial computation modulo the input ideal, and its factorization.

We revisit this algorithm, and discuss Maple and Magma implementations that contradict the somehow pessimistic conclusions of Monico's original article: this algorithm provides competitive, sometimes faster alternatives to built-in functions in both systems. We also give an estimation of the probability of success of the algorithm.

Keywords: Primary decomposition, zero-dimensional ideal, Gröbner basis, characteristic polynomial.

1 Introduction

In [9], Monico presents an algorithm for computing the primary decomposition of a zero-dimensional ideal, starting from a Gröbner basis of this ideal (see also further discussions by Cox [4, 3]). However, the experiments in [9], conducted in the Singular computer algebra system, were clearly in favor of the built-in primary decomposition routines, which implement the algorithm of [6]. A conclusion of [9] was thus that ... *"this algorithm, while relatively easy to implement, is only of practical interest if the vectorspace dimension of the quotient ring is small"*.

In this extended abstract, we report on new sets of experiments, in the Maple and Magma systems, for which our implementation of Monico's algorithm competes with, or outperforms, the built-in primary decomposition facilities.

Algorithmically, the main issue is the computation of the characteristic polynomial of an element modulo the input ideal. As in [11], we use a solution to this question relying on trace computations and Newton's formulas. Besides, Monico's algorithm is probabilistic; using classical zero-avoidance results [16, 13], we give estimates on the probability of failure of this algorithm.

2 Description of the algorithm

2.1 Overview

Let I be a zero-dimensional ideal in $k[X_1, \ldots, X_n]$, where k is any field (we denote by $\overline{k}$ one of its algebraic closures). We aim at computing a minimal primary decomposition of I, which we will write

$$I = Q_1 \cap \cdots \cap Q_s.$$

To this effect, Monico's algorithm takes as input a Gröbner basis for the ideal I, and outputs polynomials $R_1, \ldots, R_s$, such that for all $i = 1, \ldots, s$, $Q_i = I + \langle R_i \rangle$ holds. From this, one may compute Gröbner bases for all Q_i, if required.

The polynomials R_i are obtained as follows. Let A be the residue class ring

$$A = k[X_1, \ldots, X_n]/I,$$

and let t be a "generic" element in A (the genericity assumption is discussed more precisely in Subsection 2.3). Write $\chi_t \in k[T]$ for the characteristic polynomial of the endomorphism

$$\mu_t : \begin{array}{ccc} A & \to & A \\ u & \mapsto & ut, \end{array}$$

which we call the *characteristic polynomial of* t. Let finally $c_1, \ldots, c_\ell \in k[T]$ be the irreducible factors of χ_t, and $m_1, \ldots, m_\ell$ their multiplicities. For $i = 1, \ldots, \ell$, define $r_i = c_i^{m_i}(t) \in A$, and take for R_i any preimage of r_i in $k[X_1, \ldots, X_n]$. Then we have the equality (Proposition 2.3 in [9])

$$I = (I + \langle R_1 \rangle) \cap \cdots \cap (I + \langle R_\ell \rangle).$$

Furthermore, as said above, under suitable genericity conditions on t, ℓ equals the number s of primary components of I, and the ideals $I + \langle R_i \rangle$ are then these primary components.

2.2 Details of the implementations

Given the input Gröbner basis, and $t \in A$, the main tasks in this algorithm are:

1. computing the characteristic polynomial χ_t of t;
2. factoring χ_t as $\chi_t = c_1^{m_1} \cdots c_\ell^{m_\ell}$;
3. computing $c_i^{m_i}(t)$ for all i.

In this work, we concentrated on points 1 and 3, leaving factorization to the built-in routines.

Characteristic polynomial computation. Several solutions are available to compute the characteristic polynomial of an element t in A. In our experiments, the most efficient solution turned out to strongly depend on the implementation platform.

Built-in characteristic polynomial routines turned out to be the bottleneck in our preliminary Maple implementation. Our solution for this platform comes from Rouillier's work [11], and is closely related to Leverrier's algorithm [8] (note that primary decomposition is already mentioned as an application of the RUR algorithm in [11]).

Let tr be the *trace* linear form on the quotient A, which maps $t \in A$ to the trace of the map μ_t. The following classical proposition, a consequence of the so-called "Stickelberger Theorem" [5, Proposition 4.2.8] shows how to use this linear form for characteristic polynomial computation.

Proposition 2.1. *For all $t \in A$ and i in $\mathbb{N}$, $\mathrm{tr}(t^i)$ is the ith power sum (Newton sum) of the characteristic polynomial of t.*

Our algorithm for characteristic polynomial computation then follows Rouillier's. Note that as input, we know a Gröbner basis of I, and thus a monomial basis $B = b_1, \ldots, b_D$ of the quotient A. Due to the use of Newton's relations below, we have to assume that D is less than the characteristic of the base field.

1. Compute the matrix M_t of the endomorphism μ_t in the basis $b_1, \ldots, b_D$;
2. Compute the trace form, that is, the trace of all elements $b_1, \ldots, b_D$;
3. Compute the powers $1, t, \ldots, t^D$ by successively applying M_t to the vector $[1, 0, \ldots, 0]^t$;
4. Compute the traces of these vectors using the linearity of the trace;
5. Recover the polynomial χ_t using Newton's relations.

More precisely, we start by computing the multiplication matrices of all variables modulo I, and deduce the whole multiplication table of A by successive multiplications. Computing the multiplication matrix of a variable requires D reductions by the given Gröbner basis. Then, deducing the whole multiplication table requires at most $|B \times B|$ matrix/vector products in size D, hence has cost in $O(D^4)$; note however that $|B \times B|$ may be smaller than D^2. The matrix M_t, as well as the traces of all elements $b_1, \ldots, b_D$ are then deduced from the multiplication table, for $O(D^3)$ operations. Step 3 requires D matrix/vector multiplications, for a cost in $O(D^3)$; Step 4 and Step 5 have cost in $O(D^2)$. See [11] for a similar analysis.

In contrast, in the system Magma, the most efficient approach we found uses the built-in `CharacteristicPolynomial` function (with the default settings). Then, on this platform, we do not need to use explicitly the trace form, and thus, we do not need to know the whole multiplication table of A: only the matrix M_t of the multiplication map μ_t is required.

The element t is then taken as a random polynomial of degree 1: the algorithm is still valid with this restriction (see below), and the computations are substantially faster. We then determine the matrix M_t, and deduce its characteristic polynomial using built-in routines.

Evaluation. The final step of the algorithm consists in evaluating univariate polynomials at the element $t \in A$.

- In our Maple implementation, using Leverrier's idea, the required powers of t are already known, hence only constant multiplications and additions are required to perform the evaluation.
- In the Magma implementation, some more work is required, since only the multiplication matrix M_t of t is known; hence, this evaluation is done through Horner's scheme, using the matrix M_t to perform the successive multiplications by t.

2.3 Probabilistic aspects

The validity of this algorithm depends of the choice of t; we now discuss conditions that guarantee success. Recall that $Q_1 \cap \cdots \cap Q_s$ denotes a minimal primary decomposition of I.

Proposition 2.2. *Suppose that for all points α, β in $V(I)$, $t(\alpha) \neq t(\beta)$. Then the previous algorithm correctly computes the primary decomposition of I.*

Proof. By the Chinese Remainder Theorem, one may write $A = \prod_{i \leq s} A_i$, with $A_i = k[X_1, \ldots, X_n]/Q_i$. To any t in A and $i \leq s$, we associate the characteristic polynomial $\chi_{t,i}$ of the image of t in A_i; in particular, χ_t is the product of the polynomials $\chi_{t,i}$. Proposition 2.3 in [9] states that if the $\chi_{t,i}$ are pairwise coprime, the output is correct. By Stickelberger's theorem, the roots of $\chi_{t,i}$ are the values $t(\alpha)$, for α in $V(Q_i)$, and the conclusion of our proposition follows. □

From this proposition, we deduce through standard arguments that using generic elements will guarantee success. We first state a result for arbitrary elements in the quotient, and then its analogue for linear forms.

Corollary 2.3. *Let $b_1, \ldots, b_D$ be the given monomial basis of A and let $d \leq D$ be the number of distinct roots of I.*

- *There exists a non-zero polynomial $\Delta \in \overline{k}[T_1, \ldots, T_D]$ of degree $d(d-1)/2$ such that, if $\Delta(t_1, \ldots, t_D) \neq 0$, then using $t = t_1 b_1 + \cdots + t_D b_D$ yields the primary decomposition of I.*
- *There exists a non-zero polynomial $\Delta' \in \overline{k}[T_1, \ldots, T_n]$ of degree $d(d-1)/2$ such that, if $\Delta'(t_1, \ldots, t_n) \neq 0$, then using $t = t_1 X_1 + \cdots + t_n X_n$ yields the primary decomposition of I.*

Proof. Let $\alpha_1, \ldots, \alpha_d$ be the points in $V(I)$, and write $\alpha_i = \alpha_{i,1}, \ldots, \alpha_{i,n}$. We associate to α_i the linear forms

$$a_i(T_1, \ldots, T_D) = b_1(\alpha_i)T_1 + \cdots + b_D(\alpha_i)T_D \quad \text{and} \quad a'_i(T_1, \ldots, T_n) = \alpha_{i,1}T_1 + \cdots + \alpha_{i,n}T_n.$$

Finally, for i, j in $1, \ldots, d$, with $i \neq j$, we define $c_{i,j} = a_i - a_j$ and $c'_{i,j} = a'_i - a'_j$. Then

$$\Delta = \prod_{1 \leq i < j \leq d} c_{i,j} \quad \text{and} \quad \Delta' = \prod_{1 \leq i < j \leq d} c'_{i,j}$$

satisfy our requirements. □

Using the Zippel-Schwartz lemma [16, 13], we deduce the following probability estimate. We still use the notation of the previous corollary.

Corollary 2.4. *Let $\varepsilon > 0$ and let S be a subset of k of size larger than, or equal to, $d(d-1)/2\varepsilon$.*

- *Suppose that $t_1, \ldots, t_D$ are chosen uniformly at random in S. Then the probability that the algorithm outputs the correct result using $t = t_1 b_1 + \cdots + t_D b_D$ is at least $1 - \varepsilon$.*
- *Suppose that $t_1, \ldots, t_n$ are chosen uniformly at random in S. Then the probability that the algorithm outputs the correct result using $t = t_1 X_1 + \cdots + t_n X_n$ is at least $1 - \varepsilon$.*

3 Experimental results

We finally give our computation tables. Times are given in seconds, and are obtained on a 2.60GHz Pentium 4 processor with 1Gb of RAM. All computations are done modulo the prime $p = 33554393$, which leads, in view of Corollary 2.4 and of the corresponding degrees, to a probability of success of at least 0.9996. Time limits were set to 1000 seconds, and memory limits to 2Gb. Most examples below can be found on the web pages of the test suites [1, 15].

Maple implementation. We first report on the Maple implementation, made under Maple version 10. Our timings include the computation of the initial Gröbner basis, as well as those of all output primary components (first and last timing columns). Strictly speaking, these are not part of Monico's algorithm, especially concerning the final Gröbner bases computations. However, this information is obviously of interest for benchmarking this algorithm, all the more as the built-in primary decomposition routine outputs Gröbner bases for the primary components. The numbers reported in Figure 1 are then as follows:

- vars: number n of variables;
- deg: maximal degree of the input equations;
- D: dimension of the quotient A over k;
- t_1: initial Gröbner basis;

System	vars	deg	D	t_1	t_2	t_3	t_4	t_5	t_6	t_7	Total	Maple
Katsura-7	8	2	128	111	149	3	1	1.2	0.2	970	1235.4	161.2
chemkin	13	3	40	19	18	0.1	0.2	0.1	0.1	5.8	43.3	22.9
Pinchon-2	10	4	48	39.2	20.9	0.2	0.2	0.2	0.1	34.9	66.4	Error
Methan61	10	2	27	24.1	6.2	0.2	0.1	0.1	0.1	10.8	41.6	27
Rose	3	9	136	0.5	2.9	4.1	1.4	1.4	0.2	32.4	42.9	Error
Cyclic 6	6	6	156	32.5	22.1	9.7	2.2	2	0.2	182.5	251.2	48.6
4 body	6	5	138	310	167.6	5.2	1.8	1.6	0.1	441	927.3	412.3
l-4	5	3	243	0.1	2.4	51.2	4.0	7.0	0.2	314.9	379.8	Error
dessin-2	10	2	42	14.2	13.7	0.1	0.2	0.1	0.1	62.1	90.5	18.5
dessin-18-3	8	3	46	6.8	10.7	1.6	1.6	0.1	0.1	22.2	43.1	12.5
gametwo-5	5	4	44	11.8	6	0.2	0.2	0.1	0.1	15.1	33.5	15.6
r-5	5	6	121	0.1	0.7	3.7	1.0	1.0	0.1	54.1	60.7	Error

Figure 1: Maple timings modulo $p = 33554393$.

- t_2: computation of the matrices of multiplication by $X_1, \ldots, X_n$;
- t_3: computation of all multiplication matrices;
- t_4: trace computations, and deduction of the characteristic polynomial;
- t_5: factorization of the characteristic polynomial;
- t_6: evaluation of the polynomials R_i;
- t_7: Gröbner bases of all primary components;
- Total: total time, sum of $t_1, \ldots, t_7$.
- Maple: built-in Maple primary decomposition, using the function `PrimaryDecomposition` of the `PolynomialIdeals` package.

On these examples, our implementation does not quite reach the efficiency of the built-in Maple routine. However, the ratio never becomes exceedingly large, and we expect that a better tuned, lower-level implementation of Monico's algorithm would yield better results.

Remark next that on some of these examples, the built-in routine outputs the error `(in mod/GetAlgExt) only the single algebraic extension case is implemented`. Our algorithm does not require handling algebraic extensions of the base field, and thus avoids these difficulties, while preserving reasonable performances.

Note finally that the implementation relies on two distinct packages: the `Groebner` package for Gröbner bases computations, and the `LinearAlgebra:-Modular` package for all operations on matrices (using encoding of integers mod p by hardware floats). We want to underline that on all these examples, most of the time, by far, is spent on Gröbner-related operations, to compute Gröbner bases, and the matrices of multiplication by the variables $X_1, \ldots, X_n$. All other multiplication matrices are obtained by linear algebra only, which explains why they are often must faster to compute.

System	vars	deg	D	t_1	t_2	t_3	t_4	t_5	t_6	Total	Magma
Katsura-7	8	2	128	0.25	0.35	0.1	0.3	0.1	0.25	1.53	mem. > 2Gb
Katsura-8	9	2	256	2.4	4	0.15	1.2	0.1	9.4	17.25	time > 1000
chemkin	13	3	40	0.1	0.1	0.1	0.1	0.1	0.1	0.6	2.0
Pinchon-2	10	4	48	0.1	0.1	0.1	0.1	0.1	1.0	1.5	mem. > 2Gb
Methan61	10	2	27	0.1	0.1	0.1	0.1	0.1	0.1	0.6	0.5
Rose	3	9	136	0.1	0.1	0.1	0.4	0.1	0.1	0.9	3.4
Cyclic 6	6	6	156	0.2	0.1	0.1	0.1	0.1	1.0	1.6	0.3
4 body	6	5	138	0.4	0.3	0.1	0.5	0.1	1.5	1.9	mem. > 2Gb
l-4	5	3	243	0.1	0.1	0.1	0.7	0.1	2.0	3.1	3.9
dessin-2	10	2	42	0.1	0.1	0.1	0.4	0.1	0.1	0.9	165.2
dessin-18-3	8	3	46	0.1	0.1	0.1	0.1	0.1	0.1	0.6	1.4
gametwo-5	5	4	44	11.8	0.1	0.1	0.1	0.1	0.1	12.3	330.6
r-5	5	6	121	0.1	0.1	0.1	0.2	0.1	0.3	0.9	0.4

Figure 2: Magma timings modulo $p = 33554393$.

Magma implementation. Next, we discuss the results obtained using Magma version 2.11-2. As before, we count the time for computing the initial Gröbner basis, as well as Gröbner bases for all primary components. In Figure 2, all timings have been rounded up to the next integer multiple of 0.1. The legend of this figure is as follows:

- vars, deg, D: as in Figure 1;
- t_1: initial Gröbner basis;
- t_2: computation of the matrix of multiplication by t;
- t_3: computation of the characteristic polynomial;
- t_4: factorization of the characteristic polynomial;
- t_5: evaluation of the polynomials R_i;
- t_6: Gröbner bases of all primary components;
- Total: total time, sum of $t_1, \ldots, t_6$.
- Magma: built-in Magma primary decomposition, using the function `PrimaryDecomposition`.

The results on the Magma platform are more clearly in our favor, since on most examples, the built-in function takes a (sometimes much) longer time, or Magma fails by exhausting all available resources.

For these examples, the Gröbner computations are much faster in Magma than in Maple, which accounts for the large gap observed between the two systems. On the other hand, the linear algebra computations are more balanced, though still in favor of Magma. Indeed,

for both systems, linear algebra mod p is handled by similar techniques of floating-point encoding.

4 Conclusion and future work

Our main goal in this paper was to show that Monico's algorithm does actually provide a practical way to compute primary decompositions, assuming facilities for Gröbner bases computations, and univariate polynomial factorization. We substantiated this statement by means of Maple and Magma implementations, which, although implemented in a high-level environment, compete, and even outperform in some cases, the available routines. Furthermore, we stressed that the choice of implementation techniques is largely influenced by the relative efficiencies of the available tools.

Further developments are possible along these lines. Indeed, the trace computations used above are a special case of the *power projection* problem, which consists in evaluating a linear form at the powers of a given element t in A. This question, and the dual problem of evaluating a univariate polynomial at t, admit more elaborate solutions, using baby steps / giant steps techniques: the use of these techniques for evaluation is due to Paterson and Stockmeyer [10]; applying them to power projection dates to work of Kaltofen and Shoup, see [14, 7] and references therein. In our context, this baby steps / giant steps approach was already used in [2] and [12]. However, as reported in these references, making profit of this idea seems not to be immediate; more work in this direction is thus required.

References

[1] D. Bini and B. Mourrain. Polynomial test suite. `http://www-sop.inria.fr/saga/POL/`

[2] A. Bostan, B. Salvy, and É. Schost. Fast algorithms for zero-dimensional polynomial systems using duality. *Applicable Algebra in Engineering, Communication and Computing*, 14(4):239–272, 2003.

[3] D. A. Cox. Galois theory via eigenvalue methods. In D. Wang and L. Zhi, editors, *SNC 2005*, pages 166–176, 2005.

[4] D. A. Cox. Solving equations via algebras. In *Solving polynomial equations*, volume 14 of *Algorithms Comput. Math.*, pages 63–123. Springer, Berlin, 2005.

[5] D. A. Cox, J. Little, and D. O'Shea. *Using algebraic geometry*. Springer-Verlag, New York, 1998.

[6] P. Gianni, B. Trager, and G. Zacharias. Gröbner bases and primary decomposition of polynomial ideals. *Journal of Symbolic Computation*, 6(2-3):149–167, 1988. Computational aspects of commutative algebra.

[7] E. Kaltofen. Challenges of symbolic computation: my favorite open problems. *Journal of Symbolic Computation*, 29(6):891–919, 2000.

[8] U. J. J. Leverrier. Sur les variations séculaires des éléments elliptiques des sept planètes principales: Mercure, Venus, la Terre, Mars, Jupiter, Saturne et Uranus. *J. Math. Pures Appli.*, 4:220–254, 1840.

[9] C. Monico. Computing the primary decomposition of zero-dimensional ideals. *Journal of Symbolic Computation*, 34(5):451–459, 2002.

[10] M. S. Paterson and L. J. Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing*, 2(1):60–66, March 1973.

[11] F. Rouillier. Solving zero-dimensional systems through the Rational Univariate Representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.

[12] F. Rouillier. On solving zero dimensional systems with rational coefficients. Submitted to *Journal of Symbolic Computation*, 2005.

[13] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, October 1980.

[14] V. Shoup. Efficient computation of minimal polynomials in algebraic extensions of finite fields. In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (Vancouver, BC)*, pages 53–58, New York, 1999. ACM.

[15] J. Verschelde. The test database of polynomial systems.
`http://www.math.uic.edu/~jan/PHCpack/node10.html`

[16] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and algebraic computation*, number 72 in Lecture Notes in Computer Science, pages 216–226, Berlin, 1979. Springer. Proceedings EUROSAM '79, Marseille, 1979.

Marc Moreno Maza
ORCCA, University of Western Ontario, London, Ontario, Canada.
`moreno@orcca.on.ca`

Éric Schost
LIX, École polytechnique, 91128 Palaiseau, France.
`Eric.Schost@polytechnique.fr`

Wenqin Zhou
ORCCA, University of Western Ontario, London, Ontario, Canada.
`wzhou7@scl.csd.uwo.ca`

The Parametric Instability of Motion at Resonance as a Source of Chaotic Behaviour at solving a Restricted Three body Problem

Alexey E. Rosaev

Abstract

A method of dynamical system analysis is applied to a planetary restricted three-body problem (RTBP). It is well known, that equations of motion for a restricted 3-body problem in the rotating rectangular frame may be reduced to the second order differential equation with periodic coefficients (Hill's equation). Here we derive Hill's equation at the cylindrical coordinate frame. It gives the ability to estimate the width and position of instable zones. Moreover, the dependence of the position of instable zones on orbital eccentricity of the test particle is derived. It is noted, that the overlapping of instability zones in a strongly perturbed system can be a source of the chaotic behaviour.

INTRODUCTION. MAIN EQUATIONS

The three-body problem is a continuous source of study, since the discovery of its non-integrability due to H. Poincaré (1892). However, some problems are still unresolved. In particular, the explanation of the fact that in the asteroid belt some resonances are empty while others are well populated has been until recently an open problem (Celletti et al, 2002). In this paper, we consider the planar circular restricted three-body problem, when mass m revolves around M (M is much more than m) in a circular orbit and a third body is considered with negligible mass m_0 (Szebehely, 1967). We assume that all bodies move on the same plane. The equations of a planar restricted Hill's problem in the rectangular frame are:

$$\left(\frac{\partial^2}{\partial t^2}\,X\right) - 2\,m\,\left(\frac{\partial}{\partial t}\,Y\right) + f\,X = 0$$

$$\left(\frac{\partial^2}{\partial t^2}\,Y\right) + 2\,m\,\left(\frac{\partial}{\partial t}\,X\right) + g\,X = 0$$

where m - a perturbing body mass, X,Y - rectangle coordinates, f and g - known functions, t- time.
These equations may be reduced to a Hill equation for a normal distance from the variation orbit x (Szebehely, 1967). In our view, a cylindrical system has some advantages over other ones, because the variation of one of the coordinates - central distance R - is always restricted and may be considered as a small parameter at the problem.
The main equations for the planar circular 3-body problem in the absolute cylindrical frame are:

$$\left(\frac{\partial^2}{\partial t^2}\,\mathrm{R}(\,t\,)\right) - \mathrm{R}(\,t\,)\,\left(\frac{\partial}{\partial t}\,\lambda(\,t\,)\right)^2 = \frac{\partial}{\partial R}\,U$$

$$\frac{\partial}{\partial t}\,\mathrm{R}(\,t\,)^2\,\left(\frac{\partial}{\partial t}\,\lambda(\,t\,)\right) = \frac{\partial}{\partial \lambda}\,U$$

where perturbation function in the absolute frame:

$$U_0 = \frac{G\,m}{\Delta} + \frac{G\,M}{R}$$

where M- mass of the primary; m - mass of the perturbing body; R, r - distance from the mass center test particle and the perturbing body respectively; G - constant of gravity, $\delta\lambda(t) = (\omega - \omega_s)\,t + \varphi$ - angle between the perturbed and perturbing bodies(differential longitude of the perturbed body), ω, ω_s- mean motions of the perturbed and perturbing bodies respectively, φ- initial phase, and:

$$\Delta = \sqrt{R^2 + r^2 - 2\,R\,r\cos(\,\delta\,\lambda\,)}$$

There are two small parameters in the problem: x - radial shift from the intermediate (in variations) orbit and r/R - the central distances ratio of the perturbing and the perturbed bodies. Let R(t)=R+x(t), where R is constant. Accordingly, there are two ways of linearization (two possible sequences of expansion).

1 THE EQUATION IN VARIATIONS

The principle part of expansion perturbation function may be written with using Legendre's polynomials. In case of outer perturbing body (Stiefel, Scheifele, 1971):

$$U = -\frac{GM}{R} - \left(\sum_{p=2}^{\infty} \frac{Gm}{r} \left(\frac{R}{r} \right)^p P_p(\cos(\delta \lambda)) \right)$$

At linear approach by x we have:

$$\left(\frac{\partial^2}{\partial t^2} x \right) + \omega(t)^2 x = \mathrm{f}(t)$$

where f(t) and ω depend on time:

$$\omega^2 = \frac{GM}{R^3} - \frac{Gm}{\Delta^3} \left(1 - 3 \frac{R - r \cos(\delta \lambda)}{\Delta^2} \right)$$

$$\mathrm{f}(t) = \frac{{L_0}^2}{{R_0}^3} - \frac{GM}{{R_0}^2} - \left(\sum_{p=2}^{\infty} p \frac{Gm}{r^2} {y_0}^{(p-1)} P_p(\cos(\delta \lambda)) \right)$$

Then the expansion by power y= R_0/r is applied. At zero order by y:

$$\omega^2 = \frac{GM}{{R_0}^3} + \frac{1}{2} \frac{Gm}{r^3} + \frac{3}{2} \frac{Gm \cos(2 \delta \lambda)}{r^3}$$

Now we can easy take into account the eccentricity of the perturbed particle orbit and show, that such generalization gives very interesting results.
At the elliptic osculating orbit, after the introduction of osculating orbital elements a - semimajor axis, e - eccentricity, for angular momentum L we have:

$$L = R^2 \left(\frac{\partial}{\partial t} \lambda \right) = \sqrt{GM \, a \, (1 - e^2)}$$

R_0 depends on time. The phase-averaged value of R_0 is approximately:

$$\bar{R_0} = \frac{1}{2} \pi \int_0^{2\pi} \frac{a \, (1 - e^2)}{1 + e \cos(\varphi)} \, d\varphi = a \sqrt{1 - e^2}$$

After angular momentum substitution and linearization:

$$3\,\frac{{L_0}^2}{{R_0}^4} - 2\,\frac{GM}{{R_0}^3} = \frac{GM}{a^3}\,\left(1 - \frac{3}{4}\,e^4\right)$$

Finally, for the test particle elliptic motion case ω depends on the eccentricity:

$$\omega^2 = \frac{GM\,\left(1 - \frac{3}{4}\,e^4\right)}{a^3} + \left(\sum_{p=2}^{\infty} \frac{Gm}{r^3}\,p\,(\,p-1\,)\,\left(\frac{R}{r}\right)^{(\,p-2\,)}\,P_p(\,\cos(\,\delta\,\lambda\,)\,)\right)$$

Then we can group the terms with equal $p\delta\lambda$ by using Laplace's coefficients. The main frequency may be represented as follows:

$$\omega^2 = \frac{GM}{a^3}\,\left(1 - \frac{3}{4}\,e^4\right) + \left(\sum_{p=2}^{\infty} \frac{Gm}{r^3}\,b_p \cos(\,\delta\,\lambda\,)\right)$$

where b_p is easy to calculate numerically. For the periodic, or for the quasi-periodic solutions:

$$\delta\,\lambda = \sum_n \lambda_n \cos\left(n\,\omega_0\,t\right)$$

This is the Hill's equation:

$$\left(\frac{\partial^2}{\partial t^2}\,x\right) + \omega^2\,x = \mathrm{f}(\,t\,)$$

where (h is proportional to b):

$$\omega^2 = {\omega_0}^2\,\left(1 + \left(\sum_n h_n \cos\left(n\,\omega_0\,t\right)\right)\right)$$

$$\omega_0{}^2 = \frac{GM}{{R_0}^3}\,\left(1 - \frac{3}{4}\,e^4\right) = \frac{GM}{{R_0}^3}\,(\,1-\alpha\,)$$

The Hill's equation is also linear (with floating factors), so it's approximately solution possible to get a position of instability. The width of the resonance instability rapidly decreases as the resonance order increases. For the first order resonance (Landau & Lifshitz, 1971):

$$\Delta(\omega) = \frac{\omega_0 n h_n}{2n-4}$$

2 THE STABILITY OF EQUATION IN VARIATIONS SOLUTION

To investigate the stability, x may be considered as negligible small. The instability in this case results instability in general. So, we can restrict only the 1-st order in expansion by power x. The Mathieu's equation is a limit case of the Hill's equation and it is easy to study. On the other side, because the both equations (Hill's and Mathieu's) are linear, the main area of instability of the Mathieu's equation must be present in the Hill's equation solution. Hence, the areas of instability also exist in this case,and they comply with the condition:

$$\frac{\omega}{\omega_s} = \frac{nk}{nk-2+\alpha}$$

It means that the orbits near resonances (2n+1)/(2n-1) are instable parametric. This conclusion completely coincides with the results of study of the problem under discussion in the paper by Hadjidemetriou (Hadjidemetriou, 1982) by methods of matrix algebra. In the view of linear equations in an elliptic case, we can explain an interesting feature - centres of resonant zones are shifted out relative to the exact commensurability (resonance). In a simple case with one perturbing body, the centres of instable resonant zones moved away from the exact commensurabilities toward the source of perturbation. So, exact commensurability may be outside of the respective instable zone!
These expressions show, that, at strong perturbations, unstable resonance zones may be overlapped. Beginning from this point,in accordance with chaotic motion appearance,the behaviour of the system becomes completely chaotic when such zones fill all phase space.
The condition of resonance overlapping is:

$$\frac{n+1}{n-1-\frac{3}{4}e^4} - \frac{n}{n-2+\frac{3}{4}e^4} = \frac{n\,m\,b}{M\,(2n-4)\,\left(1-\frac{3}{8}e^4\right)}$$

It seems, that the width of instability increased with eccentricity. On the other hand, the distance between two nearest unstable areas $\Delta\omega$ decreased. The width of areas of instability quickly decreases with the growth of order n. However, for each m, M, the such value of n is present, since of which the overlapping of unstable zones take place (Fig.2). The resonance overlap criterion (Chiricov, 1979) affirms that whenever two resonances overlap, the corresponding resonant orbits become chaotic. For e=0, as it is evident, that overlapping

of unstable zones is possible only if limit n approaches to infinity. It means, that the effect of resonance overlapping of instable zones and related chaotic behaviour, strongly depends on the orbital eccentricity of test particle m.

3 RESULTS AND CONCLUSIONS

As a result, we obtain very suitable equation in variation to study perturbed motion at resonance.
The main result of this study may be formulated in such form:
1. There are no stable orbits with e=0 and i=0 in the vicinity of mean motion resonances (2n-1)/(2n+1).
2. Unstable zone positions and widths depend on eccentricity. Athigh order of resonance they overlap.
The results can be applied to a number of problems of the Solar system dynamics. A number of minor planets, on high elliptic orbits, fall into instable areas and show the regular behaviour of their orbits. We suppose, that stability of such orbits is provided due to large eccentricity.

References

[1] Celletti A., Chessa A., Hadjidemetriou J., Valsecchi G. B.: A Systematic Study of the Stability of Symmetric Periodic Orbits in the Planar, Circular, Restricted Three-Body Problem Celestial Mechanics and Dynamical Astronomy 83: 1-4,2002,239-255.

[2] Chiricov B.V.: A universal instability of many-dimensional oscillator systems. Phys. Rep. 52,1979,263-379.

[3] Hadjidemetriou, J.D.: On the relation between resonance and instability in planetary systems, Celestial Mechanics, vol. 27, 1982, pp. 305-322.

[4] Landau L.D.,Liphshitz E. M. Mechanics. Moscow, Nauka, 1973, p.104-105.

[5] Poincaré H.: Les Méthodes Nouvelles de la Mécanique Céleste, Gauthier Villars, Paris, 1892.

[6] Stiefel E. L., Scheifele G. Linear and Regular Celestial Mechanics. Springer-Verlag Berlin Heidelberg New York, 1971, 305 p.

[7] Szebehely, V. Theory of Orbits. The Restricted Problem of Three Bodies. Acad.press New York & London, 1967.

Rosaev A.
FGUP NPC Nedra, Yaroslavl, Russia
rosaev@nedra.ru

Obstacle to Factorization of LPDOs

Ekaterina Shemyakova Franz Winkler

Abstract

We investigate the problem of approximate factorization of linear partial differential operators of arbitrary order and in arbitrarily many variables. Given any such operator L and a specified factorization of its symbol, we define the associated ring of obstacles to the factorization of L extending the specified factorization of the symbol. We derive some facts about obstacles and give an exhaustive enumeration of obstacles for operators of order two and three.

Introduction

We consider the problem of the factorization of Linear Partial Differential Operators (LPDO) over some function space. We start with the approach of Grigoriev and Schwarz [2], who gave an algorithm for factorization of such operators with separable symbol. In each step of this algorithm one has to solve a system of linear equations, which was proved to have at most one solution. So in each step one either finds the next homogeneous component of factors in the factorization or stops and concludes that there is no factorization.

We suggest to use the information obtained by the algorithm even in the case of non-existence of any factorization, that is to describe what actually prevents a factorization. We introduce the notion of a common obstacle to factorization. The idea goes back to Laplace, who found his famous invariants as the common obstacles for second order strictly hyperbolic operators, as described in [4]. Some particular cases of this idea are considered in [3].

A common obstacle is not unique in general. However, we prove that all common obstacles belong to the same class in the ring of obstacles, which is the factor ring of the ring of differential operators modulo the homogeneous ideal generated by the factors of the symbol. We say that this class of common obstacles is the obstacle, which is defined uniquely.

The paper is organized as follows: in Section 1 we fix some notation. In Section 2, we introduce the notion of a common obstacle. Then, in Section 3, we define the ring of obstacles and the uniquely determined obstacle. Uniqueness, order estimation, and behavior of families of factorizations are investigated. In Section 4 and 5 exact formulae for all obstacles of second and third order operators are determined.

1 Notation

Let K be a commutative ring of functions with 1 in the variables $x_1, \dots, x_n$. Consider a differential ring

$$K[X_1, \dots, X_n],$$

where the differential variables $X_1, \dots, X_n$ correspond to the usual partial derivations $\partial_{x_1}, \dots, \partial_{x_n}$ respectively. We use the notation

$$X^{(i_1,\dots,i_n)} := X_1^{i_1} \dots X_n^{i_n}$$

and define the (total) order in the usual fashion:

$$|X^{(i_1,\dots,i_n)}| = ord(X^{(i_1,\dots,i_n)}) := i_1 + \dots + i_n.$$

The differential polynomial ring $K[X_1, \dots, X_n]$ is graded by the total order. The elements of $K[X_1, \dots, X_n]$ are linear partial differential operators, which we abbreviate as LPDO. Consider $L \in K[X_1, \dots, X_n]$ of order d and with the coefficients $a_J \in K,\ J \in \mathbf{N^n}$, that is

$$L = \sum_{|J| \le d} a_J X^J = \sum_{i=0}^{d} L_i, \tag{1}$$

where $L_i = \{a_J X^J \mid ord(X^J) = i\}$ is called a homogeneous component of L of order i [1]. The homogeneous component L_d is called the *symbol* of L and denoted by Sym_L.

Now, we define operations in $K[X_1, \dots, X_n]$: for two LPDOs L (1) and

$$M = \sum_{J \le r} b_J X^J$$

we define the common operation of operator composition:

$$L * M := \sum_{|I| \le d, |J| \le r} a_I X^I \left(b_J \partial^J\right),$$

the operation of the polynomial multiplication:

$$L \cdot M := \sum_{|I| \le d, |J| \le r} \left(a_I b_J X^{I+J}\right).$$

2 Common Obstacle

In this section let L be an LPDO and let its symbol Sym_L be decomposed as $Sym_L = S_1 \cdot S_2$.

Definition 2.1. We say that a factorization $L = L_1 * L_2$ such that

$$Sym_{L_1} = S_1 \quad \text{and} \quad Sym_{L_2} = S_2$$

is of the *type* $(S_1)(S_2)$ (or an extension of the factorization $Sym_L = S_1 \cdot S_2$).

Example 2.2. Consider the second order operator

$$M = (e^x + y)\partial_{xx}^2 + (x + (e^x + y)y)\partial_{xy}^2 + xy\partial_{yy}^2 + \partial_x + (x + y)\partial_y.$$

The decomposition of the symbol

$$Sym_M = (e^x + y)X^2 + (x + (e^x + y)y)XY^2 + xyY^2 = ((e^x + y)X + xY)(X + yY)$$

can be extended to the factorization

$$M = ((e^x + y)\partial_x + x\partial_y + 1) * (\partial_x + y\partial_y).$$

Remark 2.3. In general, not every decomposition of the symbol can be expanded into a factorization of the operator.

Definition 2.4. An LPDO $R \in K[X_1, \ldots, X_n]$ is called a *common obstacle* to factorization of the type $(S_1)(S_2)$ if there exists a factorization of the type $(S_1)(S_2)$ for the operator $L - R$ and R has minimal possible order.

Obviously, a common obstacle is not uniquely defined.

Remark 2.5. A common obstacle always exists, although it may be equal to 0.

Theorem 2.6. *Let L be an LPDO in two variables, $ord(L) = d$, and $Sym_L = S_1 \cdot S_2$, where S_1 and S_2 are coprime. Then the order of a common obstacle to a factorization of the type $(S_1)(S_2)$ is less or equal to $d - 2$.*

Remark 2.7. Let $deg(S_1) = deg(S_2) = 1$ and the number n of variables be 2. Then a common obstacle to factorization of the type $(S_1)(S_2)$ has order less or equal to 0. That is, any common obstacle is a zero order operator.

Remark 2.8. Let $deg(S_1) = 1$, $deg(S_2) = 2$ and the number n of variables be 2. Then a common obstacle to factorizations has order less or equal to 1.

3 Ring of obstacles

In this section let $L \in K[X_1, \ldots, X_n]$ and $Sym_L = S_1 \cdot S_2$, where S_1 and S_2 are coprime. Denote the orders of S_1 and S_2 by k and l, respectively.

Definition 3.1. We define the *ring of obstacles* as the factor ring

$$K(S_1, S_2) := K[X_1, \ldots, X_n]/\langle S_1, S_2\rangle,$$

where $\langle S_1, S_2\rangle$ is the homogeneous ideal generated by S_1 and S_2.

When L has no factorization of the type $(S_1)(S_2)$, one may, nevertheless, apply the algorithm of Grigoriev and Schwarz [2] to L, looking for a factorization of such a type. In this way, at every step one has to solve an equation in order to find the next homogeneous components of the factors of L. So, either there is a solution and we may proceed one more step, or, otherwise, we stop and have a common obstacle, which is necessarily unique by construction.

Definition 3.2. We call the common obstacle obtained by the above algorithm the *main obstacle*.

Definition 3.3. We define the *obstacle* to factorizations of the type $(S_1)(S_2)$ as the whole coset of the main obstacle in $K(S_1, S_2)$.

Theorem 3.4. *Any common obstacle belongs to the same coset in $K(S_1, S_2)$, that is the obstacle is uniquely defined as an element of the ring $K(S1, S2)$.*

Remark 3.5. So, the common obstacle is not unique, but there are the main obstacle and the obstacle, which are uniquely defined.

Remark 3.6. The factorization of L of the type $(S_1)(S_2)$ exists if and only if the obstacle equals zero. The factorization of L of the type $(S_1)(S_2)$ exists if and only if the main obstacle is zero.

Theorem 3.7. *The dimension of the ring of obstacles $K(S1, S2)$ in order $d < k + l$ is*

$$\binom{n+d-1}{n-1} - \chi(d-k)\binom{n+d-k-1}{n-1} - \chi(d-l)\binom{n+d-l-1}{n-1},$$

where

$$\chi(c) := \begin{cases} 1 & \text{if } c \geq 0\ , \\ 0 & \text{otherwise.} \end{cases}$$

Example 3.8. Let $k = l = 1$. Then by the theorem 3.7 the dimension of the ring of obstacles

- in order 0 is 1,
- in order 1 is $n - 2$.

Example 3.9. Let $k = 1$ and $l = 2$. Then by the theorem 3.7 the dimension of the ring of obstacles

- in order 0 is 1,
- in order 1 is $n - 1$,
- in order 2 is $\frac{n^2-n-2}{2}$.

Theorem 3.10. *Let f be an arbitrary function in K. Then the obstacle for the type $(S_1 \cdot \frac{1}{f})(S_2 \cdot f)$ agrees with the obstacle for the type $(S_1)(S_2)$.*

4 Obstacle for second order LPDO

In this section we assume that $n = 2$, i.e. we are in the case of two variables, and $L \in K[X_1, X_2]$ is an LPDO of second order. Suppose its symbol is decomposed as $Sym_L = S_1 \cdot S_2$, where S_1 and S_2 are coprime. Because of theorem 3.10, it is enough to find the obstacle only in the case where the coefficients in X_1 in S_1 and S_2 are 1. So we may consider

$$S_1 = X_1 + \alpha X_2, \quad S_2 = X_1 + \beta X_2,$$

where $\alpha, \beta \in K$, as the general forms of S_1 and S_2. Then the LPDO L may be written as

$$L = S_1 \cdot S_2 + a_{10}X_1 + a_{01}X_2 + a_{00},$$

for some $a_{10}, a_{01}, a_{00} \in K$. In this situation we can give an exact formula for the main obstacle of this type.

Theorem 4.1. *The main obstacle of the type* $(S_1)(S_2)$ *is*

$$c^2 - a_{10}c + a_{00},$$

where

$$c = \frac{1}{\alpha - \beta}\left(\partial_x(\beta) + \alpha\partial_y(\beta) + a_{10}\alpha - a_{01}\right).$$

Remark 4.2. The main obstacle to a factorization of the type is $(S_2)(S_1)$ is

$$c^2 - a_{10}c + a_{00},$$

where

$$c = \frac{1}{\beta - \alpha}\left(\partial_x(\alpha) + \beta\partial_y(\alpha) + a_{10}\beta - a_{01}\right).$$

Remark 4.3. So, now, from the formulae it is clear, that the obstacle to factorization of the type $(S_1)(S_2)$ is not the same as that of the type $(S_2)(S_1)$.

Remark 4.4. As mentioned at the beginning of this section, the obstacle in the case $S_1 = X_1, S_2 = X_2$ can be obtained from Theorem 4.1 by a change of variables. But for a second order operator it is easy to consider the case $S_1 = X_1, S_2 = X_2$ separately. Namely, we may find that the obstacle P_1 for the type $(X_1)(X_2)$ is

$$P_1 = a_{00} - \partial_{x_1}(a_{10}) - a_{10}a_{01}$$

and the obstacle P_2 for the type $(X_2)(X_1)$ is

$$P_2 = a_{00} - \partial_{x_2}(a_{01}) - a_{10}a_{01}.$$

Remark 4.5. One may note that the obtained obstacles P_1 and P_2 are exactly the Laplace invariants of a strictly hyperbolic second order LPDO [4] .

5 Obstacle for third order LPDO

In this section we assume that $n = 2$, i.e. we are in the case of two variables, and $L \in K[X_1, X_2]$ is an LPDO of third order. Suppose its symbol is decomposed as

$$Sym_L = S_1 \cdot S_2 \cdot S_3.$$

Because of Theorem 3.10, we may assume that the coefficients in X_1 in S_1 and S_2 are 1. So, we may say that the following are the general forms for S_1, S_2, S_3:

$$S_1 = X_1 + s_1X_2,\ S_2 = X_1 + s_2X_2,\ S_3 = X_1 + s_3X_2,$$

where $s_1, s_2, s_3 \in K$. Thus, in this section we may consider the following as the general form of L:

$$L = S_1 \cdot S_2 \cdot S_3 + L_2 + L_1 + L_0,$$

where

$$L_2 = a_{20}X_1^2 + a_{11}X_1X_2 + a_{02}X_2^2,\ L_1 = a_{10}X_1 + a_{01}X_2,\ L_0 = a_{00}$$

and all $a_{ij} \in K$.

In he following we determine the main obstacle to the factorization of L for every combination of S_1, S_2, S_3 into two factors of the form $(S_i)(S_jS_k)$ or $(S_jS_k)S_i$. It is convenient to introduce the following notation.

Definition 5.1. For $\alpha, \beta, \gamma \in K$ and $s \in \{-1, 1\}$ we define

$$\begin{aligned}
det(\alpha, \beta, \gamma, s) &:= s(\gamma - \alpha)(\beta - \alpha),\\
p_{10}(\alpha, \beta, \gamma, s) &:= \frac{1}{det(\alpha, \beta, \gamma, s)}((-\beta\gamma + \beta\alpha + \gamma\alpha)a_{20} - \alpha a_{11} + a_{02}),\\
p_{01}(\alpha, \beta, \gamma, s) &:= \frac{1}{det(\alpha, \beta, \gamma, s)}(\alpha\beta\gamma a_{20} - \beta\gamma a_{11} + (-\alpha + \beta + \gamma)a_{02}),\\
p_{00}(\alpha, \beta, \gamma, s) &:= \frac{1}{det(\alpha, \beta, \gamma, s)}(\alpha^2 a_{20} - \alpha a_{11} + a_{02}),
\end{aligned}$$

$$\begin{aligned}
P_1(\alpha, \beta, \gamma) &:= (a_{10} - S_1(p_{10}(\alpha, \beta, \gamma, s)) + g_{00}p_{01}(\alpha, \beta, \gamma, s)) \cdot X_1 +\\
&\quad (a_{01} + S_1(p_{01}(\alpha, \beta, \gamma, s)) + g_{00}p_{01}(\alpha, \beta, \gamma, s)) \cdot X_2 + a_{00},\\
P_2(\alpha, \beta, \gamma) &:= a_{10}X_1 + (a_{01} - (S_1 \cdot S_2 + p_{10}(\gamma, \beta, \alpha, 1)X_1 +\\
&\quad p_{01}(\gamma, \beta, \alpha, 1)X_2)(\gamma))X_2 + a_{00} - (S_1 \cdot S_2)(p_{00}(\gamma, \beta, \alpha, 1)).
\end{aligned}$$

Remark 5.2. Note that in the $p_I(\alpha, \beta, \gamma, s),\ I \in \{(10), (01), (00)\}$ and $P_1(\alpha, \beta, \gamma)$ the second and the third variables commute, while in $P_2(\alpha, \beta, \gamma)$ the first and the second variables commute.

Theorem 5.3. *For the types $(S_i)(S_j \cdot S_k)$ and $(S_j \cdot S_k) \cdot (S_i)$, where S_i is coprime with $S_j \cdot S_k$, the main obstacles are $P_1(s_i, s_j, s_k)$ and $P_2(s_j, s_k, s_i)$, respectively.*

Acknowledgement

We acknowledge support for this work from FWF (Austrian national science fund) under the projects SFB F013/F1304 and P16357-N04.

References

[1] J. Björk. Rings of differential operators, North-Holland Publishing Commpany, Amsterdam - Oxford - New York. 1979.

[2] D.Grigoriev, F.Schwarz. Factoring and Solving Linear Partial Differential Equations, J. Computing 73, pp.179-197. 2004.

[3] E. Kartashova. Hierarchy of general invariants for bivariate LPDOs. to appear in J. Theor.Math.Phys., May 2006.

[4] S.P. Tsarev. Generalized Laplace Transformations and Integration of Hyperbolic Systems of Linear Partial Differential Equations, Proc. ISSAC'05, 325-331. 2005.

Ekaterina Shemyakova, Franz Winkler
Research Institute for Symbolic Computations (RISC)
J.Kepler University, Altenbergerstr. 69, A-4040 Linz, Austria
{kath, Franz.Winkler}@risc.uni-linz.ac.at
http://www.risc.uni-linz.ac.at/

Fraction-free forms of LU matrix factoring

Wenqin Zhou David J. Jeffrey Robert M. Corless

Abstract

This paper considers LU matrix factoring. It addresses the, essentially practical, question of how to represent the results of fraction-free algorithms (also called exact-division) in a fraction-free form. Thus, at present, a system such as MAPLE can perform fraction-free calculations, but cannot return fraction-free matrices because there is no widely accepted form in which to do so. This paper reviews one previous attempt to define such a form and presents a new form that is more convenient.

1 Introduction

Developers of computer algebra systems are sometimes frustrated by the lack of standard mathematical forms in which to return their results. Matrix LU factoring is an example. The terms *fraction free* and *exact division* have each been used to describe methods used in linear algebra. Bareiss [1] used *fraction free* for his version of Gaussian elimination (which he credited to Jordan), while Erlingsson, Kaltofen & Musser [2] used *exact division* to describe their version of Gram—Schmidt orthogonalization, which they credit to [3]. Given the existence of two apparently equivalent terms, we propose using the term *exact division* to describe an algorithm that relies on divisions being exact, and the term *fraction free* to describe any output form which contains only entries in the input domain. In other words, one term describes the process and the other the result. (The term *division-free* is also available.)

A fraction-free output need not come from an exact-division algorithm. Thus, although here we consider Bareiss-type algorithms, one could equally consider computations based on other techniques, such as p-adic lifting [8]. Further, the fraction-free idea is not confined to linear algebra: below, we shall mention pseudo-division of polynomials.

Turing's description of Gaussian elimination as a matrix factoring [4] is well established in linear algebra, particularly numerical linear algebra [5]. Even though the factoring takes a number of different forms (Cholesky, Doolittle-LU, Crout-LU, Turing), only relatively recently has there been an attempt to define a fraction-free form [6, 7]. Thus at present MAPLE offers LU factoring with the option `FractionFree`, but the output contains fractions.

```
> LUDecomposition(<<10,9>|<7,6>>,method='FractionFree');
```

$$\left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right], \left[\begin{array}{cc} 1 & 0 \\ \frac{9}{10} & \frac{1}{10} \end{array}\right], \left[\begin{array}{cc} 10 & 7 \\ 0 & -3 \end{array}\right]$$

Thus a user cannot continue a fraction-free calculation from this point without first undoing the divisions that MAPLE has done. The problem is that all forms of Turing's factoring have been defined primarily in a numerical context, and the special considerations of exact computation have been neglected.

A fraction-free form for LU factoring was given in [6], where the following theorem was stated. Although stated for matrices over $\mathbb{Z}$, it clearly applies to more general input domains.

Theorem 1.1. *The matrix $A \in \mathbb{Z}^{n\times n}$ may be written*

$$F_1 P A = L F_2 U \ , \tag{1}$$

where $F_1 = \mathrm{diag}(1, p_1, p_1p_2, \ldots, p_1p_2\cdots p_{n-1})$, P is a permutation matrix, $L \in \mathbb{Z}^{n\times n}$ is unit lower triangular, $F_2 = \mathrm{diag}(1, 1, p_1, p_1p_2, \ldots, p_1p_2\cdots p_{n-2})$, and $U \in \mathbb{Z}^{n\times n}$ is upper triangular. The p_i are the pivots that arise.

Proof. A proof is given in [6], and is not reproduced. □

This factoring is modelled on other fraction-free definitions, such as pseudo-division of polynomials. There, polynomials $a, b \in \mathbb{Z}[x]$ are divided by writing [9]

$$\beta a(x) = b(x)q(x) + r(x) \ ,$$

where β inflates the coefficients of $a(x)$ so as to guarantee that $q(x), r(x) \in \mathbb{Z}[x]$. Analogously, the matrix F_1 above serves to inflate the matrix A so that L, U are fraction free. However, although this model is satisfactory for pseudo-division, in the matrix case it leads to two unsatisfactory features: first, two inflating matrices are required, and secondly, the matrices are clumsy, containing entries that increase rapidly in size. If the model of pseudo-division is abandoned, then a tidier factoring is possible. Again, we state the theorem for $\mathbb{Z}$ with the generalization being clear.

Theorem 1.2. *The matrix $A \in \mathbb{Z}^{n\times n}$ may be written*

$$\begin{aligned} PA &= LD^{-1}U \\ L &= \begin{bmatrix} p_1 & & & & \\ l_{21} & p_2 & & & \\ \vdots & \vdots & \ddots & & \\ l_{n-1,1} & l_{n-1,2} & \cdots & p_{n-1} & \\ l_{n1} & l_{n2} & \cdots & \cdots & 1 \end{bmatrix}, \quad U = \begin{bmatrix} p_1 & u_{12} & \cdots & \cdots & u_{1n} \\ & p_2 & \cdots & \cdots & u_{2n} \\ & & \ddots & \vdots & \vdots \\ & & & p_{n-1} & u_{n-1,n} \\ & & & & p_n \end{bmatrix} \\ D &= \mathrm{diag}(p_1, p_1p_2, \ldots, p_{n-2}p_{n-1}, p_{n-1}) \ . \end{aligned} \tag{2}$$

where P is a permutation matrix, L and U are triangular as shown, and the p_i are the pivots arising in the Gaussian elimination. The pivot p_n is also the determinant of matrix A.

Proof. The proof is based on the Bareiss procedure, but as stated above, there is no implication that this procedure is the only way to compute the factoring. The proof is by induction. If A_k is a $k \times k$ matrix, then we start at $k = 3$.

$$A_3 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22}^{(1)} & 0 \\ a_{31} & a_{32}^{(1)} & 1 \end{bmatrix} \begin{bmatrix} p_1 & 0 & 0 \\ 0 & p_1 p_2 & 0 \\ 0 & 0 & p_2 \end{bmatrix}^{-1} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} \\ 0 & 0 & a_{33}^{(2)}/a_{11} \end{bmatrix} .$$

Here $p_1 = a_{11}$, $p_2 = a_{22}^{(1)}$, $a_{ij}^{(1)} = a_{ij}a_{11} - a_{i1}a_{1j}$ and $a_{33}^{(2)} = a_{33}^{(1)}a_{22}^{(1)} - a_{23}^{(1)}a_{32}^{(1)}$, implying

$$a_{33}^{(2)}/a_{11} = a_{33}a_{11}a_{22} - a_{33}a_{21}a_{12} - a_{31}a_{13}a_{22} - a_{32}a_{11}a_{23} + a_{32}a_{21}a_{13} + a_{31}a_{12}a_{23} ,$$

where the well-known exact-divisibility is seen. Now suppose the theorem is true for A_k. For convenience, we label the elements of the matrix from 2 to $k+1$, and using the obvious block notation, we write

$$A_k = \begin{bmatrix} a_{22} & \mathbf{a}_{23} \\ \mathbf{a}_{32} & \mathbf{A}_{33} \end{bmatrix} = \begin{bmatrix} p_2 & \\ \mathbf{a}_{32} & \hat{\mathbf{L}} \end{bmatrix} \begin{bmatrix} p_2 & \\ & \hat{\mathbf{D}} \end{bmatrix}^{-1} \begin{bmatrix} p_2 & \mathbf{a}_{23} \\ & \hat{\mathbf{U}}_{33} \end{bmatrix} .$$

Now for A_{k+1} we have

$$\begin{aligned} A_{k+1} &= \begin{bmatrix} a_{11} & a_{12} & \mathbf{a}_{13} \\ a_{21} & a_{22} & \mathbf{a}_{23} \\ \mathbf{a}_{31} & \mathbf{a}_{32} & \mathbf{A}_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & & \\ a_{21} & 1 & \\ \mathbf{a}_{31} & 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} p_1 & & \\ & p_1 & \\ & & p_1\mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} a_{11} & a_{12} & \mathbf{a}_{13} \\ & a_{22}^{(1)} & \mathbf{a}_{23}^{(1)} \\ & \mathbf{a}_{32}^{(1)} & \mathbf{A}_{33}^{(1)} \end{bmatrix} \\ &= \begin{bmatrix} a_{11} & & \\ a_{21} & 1 & \\ \mathbf{a}_{31} & 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} p_1 & & \\ & p_1 & \\ & & p_1\mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & a_{22}^{(1)} & 0 \\ 0 & \mathbf{a}_{32}^{(1)} & \hat{\mathbf{L}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & p_2 & 0 \\ 0 & 0 & \hat{\mathbf{D}} \end{bmatrix}^{-1} \begin{bmatrix} a_{11} & a_{12} & \mathbf{a}_{13} \\ 0 & a_{22}^{(1)} & \mathbf{a}_{23}^{(1)} \\ 0 & 0 & \hat{\mathbf{U}}^{(1)} \end{bmatrix} \\ &= \begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22}^{(1)} & 0 \\ \mathbf{a}_{31} & \mathbf{a}_{32}^{(1)} & \hat{\mathbf{L}} \end{bmatrix} \begin{bmatrix} p_1 & 0 & 0 \\ 0 & p_1 p_2 & 0 \\ 0 & 0 & \hat{\mathbf{D}} \end{bmatrix}^{-1} \begin{bmatrix} a_{11} & a_{12} & \mathbf{a}_{13} \\ 0 & a_{22}^{(1)} & \mathbf{a}_{23}^{(1)} \\ 0 & 0 & \hat{\mathbf{U}}/p_1 \end{bmatrix} \end{aligned}$$

Since $\hat{\mathbf{U}}$ is the same as $\mathbf{A}_{33}^{(2)}$, the division is exact, a fact known to Camille Jordan [1].

□

2 Concluding remarks

In [7] there is a discussion of fraction-free forward and backward substitution, which complements the LU factoring itself. Lack of space prevents such a discussion here. An anonymous reviewer has offered an attractive alternative to the discussion of theorem 1.2, which we have not used because we do not know the reviewer's identity. Analogous results can be obtained for QR factoring, but space limitations prevent us giving details.

References

[1] Erwin H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of Computation*, Vol. 22, No. 103, page 565-578, 1968.

[2] Úlfar Erlingsson, Erich Kaltofen & David Musser. Generic Gram—Schmidt Orthogonalization by Exact Division. *International Symposium on Symbolic and Algebraic Computation, ACM*, page 275-282, 1996.

[3] A.K. Lenstra, H.W. Lenstra & L. Lovász. Factoring polynomials with rational coefficients, Math. Ann. **261**, 1982.

[4] Alan M. Turing. Rounding-off errors in matrix processes. *Quart. J. Mech. Appl. Math.*, Vol. 1, page 287-308, 1948.

[5] Lloyd N. Trefethen & David Bau III. Numerical Linear Algebra, SIAM 1997.

[6] R.M. Corless & D.J. Jeffrey. The Turing factorization of a rectangular matrix, *SIGSAM Bull., ACM Press*, Vol. 31, No. 3, page 20-30, 1997.

[7] G.C. Nakos, P.R. Turner & R.M. Williams. Fraction-free algorithms for linear and polynomial equations. *SIGSAM Bull., ACM Press*, Vol. 31, No. 3, pp 11-19, 1997.

[8] John D. Dixon. Exact solution of linear equations using p-adic expansions. *Numerische Mathematik*, Vol 40, pp. 137-141, 1982.

[9] K.O. Geddes, G. Labahn & S. Czapor. Algorithms for Computer Algebra, *Kluwer*, 1992.

Wenqin Zhou
Ontario Research Centre for Computer Algebra, and Department of Applied Mathematics
The University of Western Ontario, London, Ontario, Canada N6A 5B7
wzhou7@uwo.ca
http://publish.uwo.ca/~wzhou25/

D. J. Jeffrey
Department of Applied Mathematics, and Ontario Research Centre for Computer Algebra
The University of Western Ontario, London, Ontario, Canada N6A 5B7
djeffrey@uwo.ca
http://www.apmaths.uwo.ca/~djeffrey/

R. M. Corless
Ontario Research Centre for Computer Algebra, and Department of Applied Mathematics
The University of Western Ontario, London, Ontario, Canada N6A 5B7
Rob.Corless@uwo.ca
http://www.apmaths.uwo.ca/~rcorless/

Author index