Contents

Automated Handling and Proving of Geometric Theorems

Dongming Wang

School of Science, Beihang University
Beijing 100083, China
LIP6-UPMC-CNRS, 8 rue du Capitaine Scott
75015 Paris, France
`http://www-calfor.lip6.fr/~wang`

In this talk, I will first give a brief review on the history, recent developments, and current state of automated geometric reasoning, with emphasis on the role of computer algebra for theorem proving. I will then discuss how geometric theorems may be specified, manipulated, and proved automatically. The discussed aspects will be illustrated with live examples running in my GEOTHER environment. The problem of designing and implementing a geometric-object-oriented language for symbolic geometric computing and reasoning will be addressed.

Differential Rational Normal Forms and Representations of Hyperexponential Functions (extended abstract)

Ha Le and Ziming Li

ha.le@inria.fr
Algorithms Project
INRIA Rocquencourt
78153 Le Chesnay Cedex, France
z6li@scg.math.uwaterloo.ca
Symbolic Computation Group
University of Waterloo
Waterloo, Canada

Summary. We describe differential rational normal forms of a rational function and their properties. These normal forms are used as the basis for algorithms which solve the multiplicative and additive decomposition problems of hyperexponential functions.

1 Introduction

Let $\mathbb{F}$ be a field of characteristic zero. A nonzero function $T(x)$ over $\mathbb{F}$ is hyperexponential if the logarithmic derivative $R(x) = T'(x)/T(x)$ is a rational function of x. The rational function R is called the *certificate* of T.

Representations of $R \in \mathbb{F}(x)$ in the form

$$R(x) \;=\; K(x) + \frac{S'(x)}{S(x)} \tag{1}$$

where $K, S \in \mathbb{F}(x)$ satisfy some specific conditions play a key role in a number of computer algebra algorithms operating on hyperexponential functions. Gosper's algorithm for hyperexponential indefinite integration and Zeilberger's algorithm for hyperexponential definite integration [3] both start with the certificate of a hyperexponential function. Each algorithm then proceeds by representing this certificate in the form (13). The algorithms for computing the multiplicative and additive decompositions of hyperexponential functions also use this representation.

By using the certificate R of $T(x)$, we can write

$$T(x) = \exp\left(\int R(x)\,dx\right). \tag{2}$$

Let $K(x)$ in (13) be written as $a(x)/b(x)$ where $a, b \in \mathbb{F}[x]$ and $\gcd(b, a - ib') = 1$ for all $i \in \mathbb{Z}$. Then (13) is a differential rational normal form (DRNF) of $R(x)$. By using any DRNF of R, we can rewrite (2) in the form

$$T(x) = S(x)\exp\left(\int K(x)\,dx\right). \tag{3}$$

A representation of $T(x)$ in the form (3) is called a *multiplicative decomposition* of $T(x)$.

The paper is organized as follows. In Sections 2 and 3 we define the notion of DRNFs of a rational function, and provide an algorithm for constructing them. The construction is based on a classification and distribution of the simple fractions in the irreducible partial fraction decomposition of the input rational function. These DRNFs can be considered as the differential analogue of the RNFs in the difference case [2, 1]. Among all DRNFs of a rational function, we select two canonical forms (DRCFs). While DRCF_1 helps determine similarity between two hyperexponential functions [6], DRCF_2 is used in the algorithm for solving the additive decomposition problem (also known as the reduction algorithm) for hyperexponential functions [6]. In Section 4, we show how to construct a multiplicative decomposition of a hyperexponential function based on a DRNF of its certificate, and specify the problem of computing an additive decomposition of hyperexponential functions. This problem is solved in [6]. In Section 5 we describe an implementation of the algorithms in this paper, and its availability.

For $R \in \mathbb{F}(x)$, $\mathrm{num}(R)$ and $\mathrm{den}(R)$ denote the numerator and the denominator of R, respectively. Except mentioned otherwise, we assume that $\mathrm{num}(R)$ and $\mathrm{den}(R)$ are co-prime, and $\mathrm{den}(R)$ is monic. In particular, $\mathrm{num}(0) = 0$ and $\mathrm{den}(0) = 1$. The use of some technical terms is borrowed from [2].

2 DRNFs and their strict versions

Definition 1 *A rational function $R \in \mathbb{F}(x)$ is* differential-reduced *if*

$$\gcd(\mathrm{den}(R), \mathrm{num}(R) - i\,\mathrm{den}(R)') = 1 \quad \textit{for all integers } i.$$

In the differential Gosper's algorithm(see [3]) a rational function R is constructed such that

$$\gcd(\mathrm{den}(R), \mathrm{num}(R) - i\,\mathrm{den}(R)') = 1 \quad \text{for all nonnegative integers } i.$$

The definition of differential-reduced rational functions is more restrictive so that certain denominators of rational functions in our reduction algorithm for hyperexponential functions will have minimal degree [6].

Definition 2 *Let $R \in \mathbb{F}(x)$. If there are $K, S \in \mathbb{F}[x]$ such that*

(i) $R = K + \dfrac{S'}{S}$,
(ii) *the rational function K is differential-reduced,*

then (K, S) is a differential rational normal form (DRNF) *of R. Additionally, if*

(iii)$\gcd(\mathrm{den}(R), \mathrm{den}(S)) = 1$,

then (K, S) is a strict DRNF of R. The rational functions K and S are called, respectively, the kernel *and the* shell *of the DRNF (K, S).*

A rational function R can be uniquely written as

$$R = p + \sum_{i=1}^{n} \sum_{j=1}^{m_i} \frac{q_{ij}}{d_i^j}, \tag{4}$$

where n, m_i are nonnegative integers, the d_i's are distinct, monic and irreducible polynomials in $\mathbb{F}[x]$, the polynomial q_{ij} is of degree less than that of d_i, and p is a polynomial. We call (4) the irreducible partial fraction decomposition of R over $\mathbb{F}$. By a simple fraction we mean either a polynomial or a fraction whose denominator is a power of a square-free polynomial b of positive degree, and whose numerator is of degree less than $\deg b$. All fractions appearing in (4) are simple. An easy calculation shows

Lemma 1 *A rational function is a logarithmic derivative of some rational function in $\mathbb{F}(x)$ if and only if its irreducible partial fraction decomposition can be written as $\sum_i n_i p_i'/p_i$ where the n_i's are nonzero integers and the p_i's are irreducible polynomials in $\mathbb{F}[x]$.*

The following lemma describes a relation between a differential-reduced rational function and its irreducible partial fraction decomposition.

Lemma 2 *A rational function $R(x)$ is differential-reduced iff, for any monic and irreducible p and nonzero integer m, the appearance of $\frac{mp'}{p}$ in the irreducible partial fraction decomposition of R implies that p^2 divides* $\mathrm{den}(R)$.

Proof: Set $a = \mathrm{num}(R)$, $b = \mathrm{den}(R)$. Suppose that the pair (a, b) is differential-reduced, and that $\frac{mp'}{p}$ appears in the irreducible partial fraction decomposition, but that p^2 does not divide b. Then the irreducible partial fraction decomposition of $\frac{a}{b}$ would be written as

$$\frac{a}{b} = g + \frac{mp'}{p} + \sum_i \sum_j \frac{q_{ij}}{d_i^j}$$

where $g, q_{ij}, d_i \in \mathbb{F}[x]$ and $\gcd(d_i, p) = 1$ for all i. Thus

$$\frac{a}{b} = \frac{u}{v} + \frac{mp'}{p} = \frac{up + mp'v}{vp}$$

where $u, v \in \mathbb{F}[x]$, $\gcd(u, v) = 1$, and $\gcd(p, v) = 1$. Since $\gcd(vp, up+mp'v) = 1$, we have $a = up + mp'v$ and $b = vp$. However, we can verify that p divides $\gcd(b, a - mb')$ by a direct calculation, a contradiction.

Conversely, suppose that $g = \gcd(b, a - mb')$ is of positive degree. Let p be an irreducible factor of g. Then p^2 does not divide b, for otherwise, p would divide a, a contradiction to $\gcd(a, b) = 1$. Thus

$$\frac{a}{b} = \frac{u}{v} + \frac{q}{p} \tag{5}$$

where $\gcd(u, v) = 1$, $\gcd(v, p) = 1$ and $\deg q < \deg p$. It follows that $a = up + qv$ and $b = vp$. So

$$a - mb' = v(q - mp') + (u - mv')p.$$

Hence p divides $v(q - mp')$, so it divides $(q - mp')$. A degree argument implies that $q = mp'$. Hence $\frac{mp'}{p}$ appears in the irreducible partial fraction decomposition of a/b. ■

By Lemma 2 a rational function is differential-reduced if and only if it has no first-order pole with integral residue.

Consider the irreducible partial fraction decomposition of a rational function

$$R = \sum_i \frac{u_i(x)}{v_i(x)}. \tag{6}$$

Each simple fraction u_i/v_i in (6) belongs to one and only one of the following three classes:

(I) $\frac{u_i}{v_i} = m_i \frac{v_i'}{v_i}$, $m_i \in \mathbb{Z} \setminus \{0\}$, v_i^2 does not divide $\mathrm{den}(R)$;
(II) $\frac{u_i}{v_i} = m_i \frac{v_i'}{v_i}$, $m_i \in \mathbb{Z} \setminus \{0\}$, v_i^2 divides $\mathrm{den}(R)$;
(III) $\frac{u_i}{v_i}$ is not a logarithmic derivative of any rational function.

Let (K, S) be a DRNF of R. Then

- The simple fractions in class (I) appear in the irreducible partial fraction decomposition of S'/S, not in the irreducible partial fraction decomposition of K (otherwise, K is not differential-reduced);
- The simple fractions in class (III) appear in the irreducible partial fraction decomposition of K, not in the irreducible partial fraction decomposition of K'/K (otherwise, S'/S would not be a logarithmic derivative of any rational function);
- The simple fractions in class (II) can appear in the irreducible partial fraction decomposition of either K or S'/S.

Lemma 3 *For nonzero $R \in \mathbb{F}(x)$, let u_i/v_i be a simple fraction of class (I) or (II), then v_i is an irreducible polynomial in $\mathbb{F}[x]$.*

Proof: Since u_i/v_i is of class (I) or (II),

$$\frac{u_i}{v_i} = m_i \frac{v_i'}{v_i} \tag{7}$$

for some $m_i \in \mathbb{Z} \setminus \{0\}$. If v_i is not irreducible, then $v_i = p^s$ where p is irreducible, and $s > 1$. By (7), $u_i/v_i = m_i\, s\, p'/p$. Since $\deg p < \deg v_i$, $\gcd(u_i, v_i)$ is not trivial, a contradiction. ■

The following corollary is immediate from Lemma 3.

Corollary 1 *Let the irreducible partial fraction decomposition of nonzero $R \in \mathbb{F}(x)$ be of the form (6). For $i \neq j$, if the simple fractions u_i/v_i is of class (I) and u_j/v_j is of either class (II) or class (III), then* $\gcd(v_i, v_j) = 1$.

The next lemma reveals a relation among different DRNF's.

Lemma 4 *For nonzero $R_1, R_2 \in \mathbb{F}(x)$, let (K_1, S_1) and (K_2, S_2) be two DRNFs of R_1 and R_2. If*

$$R_1 - R_2 = \frac{R'}{R}, \quad \textit{for some nonzero } R \in \mathbb{F}(x), \tag{8}$$

then $\mathrm{den}(K_1) = \mathrm{den}(K_2)$*. In particular, the kernels of all DRNF's of a rational function have the same denominator.*

Proof: Equation (8) implies

$$K_1 - K_2 = \frac{Q'}{Q} \quad \text{for some nonzero } Q \in \mathbb{F}(x), \tag{9}$$

Let p be an irreducible factor of $\mathrm{den}(K_1)$ with multiplicity m. Then a simple fraction $\frac{q}{p^m}$ must appear in the irreducible partial fraction decomposition of K_1. If $m > 1$, then there is a simple fraction $\frac{q}{p^m}$ appearing in the irreducible fraction of K_2, because all simple fractions in the irreducible partial fraction decomposition of Q'/Q have square-free denominators by Lemma 1. If $m = 1$, then $q \neq ip'$ for any integer i, for, otherwise, K_1 is not differential-reduced by Lemma 2. It follows from (9) that there exists a simple fraction f/p in the irreducible partial fraction decomposition of K_2 such that the difference of q/p and f/p is a logarithmic derivative of some rational function. Therefore p^m is also a factor of $\mathrm{den}(K_2)$. Consequently, $\mathrm{den}(K_1)$ divides $\mathrm{den}(K_2)$. In the same way we have $\mathrm{den}(K_2)$ divides $\mathrm{den}(K_1)$. Hence $\mathrm{den}(K_1) = \mathrm{den}(K_2)$ since they are monic.

Setting $R = 1$ in (8) yields the last conclusion of the lemma. ■

Example 1 Consider the rational function

$$R = \frac{4}{x-2} + \frac{4}{x+1} - \frac{3}{(x+1)^2} - \frac{9}{(x-1)^2} - \\ \frac{9x^2+12}{x^3+4x-2} + \frac{1}{(x^3+4x-2)^2}.$$

Note that R is already in the irreducible partial fraction form. The simple fractions of R are classified as follows:

$$\text{(I)}\ \ u_1 = \frac{4}{x-2}, \qquad \text{(II)}\ \ v_1 = \frac{4}{x+1},\ v_2 = -\frac{9x^2+12}{x^3+4x-2},$$

$$\text{(III)}\ \ w_1 = -\frac{9}{(x-1)^2},\ w_2 = -\frac{3}{(x+1)^2},\ w_3 = \frac{1}{(x^3+4x-2)^2}.$$

Now we construct four different DRNF's of R.
The first DRNF is constructed by moving both simple fractions in class (II) to the shell:

$$\left(w_1 + w_2 + w_3,\ \frac{\mathrm{den}(u_1)^4 \mathrm{den}(v_1)^4}{\mathrm{den}(v_2)^3}\right).$$

The second DRNF is constructed by moving both simple fractions in class (II) to the kernel:

$$\left(w_1 + w_2 + w_3 + v_1 + v_2,\ \mathrm{den}(u_1)^4\right).$$

The third DRNF is constructed by moving the simple fraction v_1 to the shell and the simple fraction v_2 to the kernel:

$$\left(w_1 + w_2 + w_3 + v_2,\ \mathrm{den}(u_1)^4 \mathrm{den}(v_1)^4\right).$$

Finally, the fourth DRNF is constructed by moving the simple fraction v_2 to the shell and the simple fraction v_1 to the kernel:

$$\left(w_1 + w_2 + w_3 + v_1,\ \frac{\mathrm{den}(u_1)^4}{\mathrm{den}(v_2)^3}\right).$$

The set of strict DRNFs of R consists of the second and the third DRNFs.

3 Two differential rational canonical forms

Among all possible DRNFs of a non-zero rational function $R(x)$, we select two differential rational canonical forms (DRCF). The selection is based on the distribution of the simple fractions of class (II) to either the kernel K or the shell S: all simple fractions of class (II) are moved to the shell for DRCF$_1$, and to the kernel for DRCF$_2$.

Example 2 For the rational function $R(x)$ in Example 1, the first DRNF is DRCF$_1$ of R, while the second DRNF is DRCF$_2$ of R.

The next theorem shows the minimality of the shell of the DRCF.

Theorem 1 *For $R \in \mathbb{F}(x)$, let S be the shell of the DRCF of R, and $\tilde{S}$ be the shell of any DRNF of R. Then* $\operatorname{den}(S)$ *divides* $\operatorname{den}(\tilde{S})$, $\operatorname{num}(S)$ *divides* $\operatorname{num}(\tilde{S})$.

Proof: Let R be of the form (6), A and B be the sets of simple fractions of class (I) and class (II), respectively. Each element f of either A or B is of the form $m\,v'/v$ where v is monic, and irreducible in $\mathbb{F}[x]$, and m, denoted by $\operatorname{res}(f)$, is a nonzero integer. Then

$$S = \prod_{f\in A} \operatorname{den}(f)^{\operatorname{res}(f)}, \quad \tilde{S} = S \underbrace{\prod_{g\in J} \operatorname{den}(g)^{\operatorname{res}(g)}}_{W} \tag{10}$$

where J is a subset of B. By Corollary 1, $\operatorname{den}(f)$ and $\operatorname{den}(g)$ are co-prime. Hence, $\operatorname{num}(\tilde{S}) = \operatorname{num}(S)\operatorname{num}(W)$, $\operatorname{den}(\tilde{S}) = \operatorname{den}(S)\operatorname{den}(W)$. ■

Corollary 1 and the first equality of (10) imply

Corollary 2 *If (K, S) is the DRCF of a rational function,* $\operatorname{den}(K)$, $\operatorname{num}(S)$ *and* $\operatorname{den}(S)$ *are pairwise co-prime.*

It follows from Corollary 2 that DRCF$_2$ is a strict DRNF, while DRCF$_1$ in general is not. Note that DRCF$_2$ of a rational function is briefly mentioned in [4, Chap. 8].

We conclude this section by describing algorithm DRCF which computes DRCF$_1$ and DRCF$_2$ of a rational function. The algorithm needs an auxiliary function for computing the class to which a simple fraction belongs.

```
Algorithm WhichClass
input:   $R \in \mathbb{F}(x) \setminus \{0\}$, a simple fraction $q_{ij}/d_i^j$ of $R$;
output:  $(1, m)$ if $q_{ij}/d_i^j$ is of class (I), $m \in \mathbb{Z} \setminus \{0\}$ such that
             $q_{ij}/d_i^j = m\left(d_i^j\right)'/d_i^j$.
         $(2, m)$ if $q_{ij}/d_i^j$ is of class (II), $m \in \mathbb{Z} \setminus \{0\}$ such that
             $q_{ij}/d_i^j = m\left(d_i^j\right)'/d_i^j$.
         $(3, 0)$ if $q_{ij}/d_i^j$ is of class (III).

  $m := q_{ij}/\left(d_i^j\right)'$;
  if $m \in \mathbb{Z} \setminus \{0\}$ then
      if $d_j^{2j}$ does not divide $\operatorname{den}(R)$ then return $(1, m)$;
      else return $(2, m)$;
      fi;
  else return $(3, 0)$;
  fi;
```

`Algorithm` **DRCF[h]**
`Input:` $R \in \mathbb{F}(x) \setminus \{0\}$ of the form (4), $h \in \{1, 2\}$;
`Output:` DRCF$_1$ of R if $h = 1$, and DRCF$_2$ of R otherwise.

```
$K := p$; $S := 1$;
for each simple fraction $q_{ij}/d_i^j$ do
    $(c, m) :=$ *WhichClass* $(R, q_{ij}/d_i^j)$;
    if $c = 1$ then $S := S \cdot d_i^{jm}$;
    elif $c = 2$ then $K := K + q_{ij}/d_i^j$;
    else if $h = 1$ then $S := S \cdot d_i^{jm}$;
        else $K := K + q_{ij}/d_i^j$;
        fi;
    fi;
od;
return $(K,\ S)$.
```

4 Representations of hyperexponential functions

4.1 Multiplicative decompositions

Definition 3 *Let $T(x)$ be a hyperexponential function over $\mathbb{F}$ with the certificate $R \in \mathbb{F}(x)$. Let the pair of rational functions (K, S) be a DRNF of R. Then T can be written as*

$$T(x) = S(x) \exp\left(\int K(x)\,dx\right),$$

which is called a multiplicative decomposition *of T.*

The following is a description of the algorithm which constructs two multiplicative decompositions of T based on the two DRCF's of R.

`Algorithm` **DMD[i]**
`input:` a hyperexponential function $T(x)$, $i \in \{1,\ 2\}$
`output:` a multiplicative decomposition $S(x) \exp\left(\int K(x)\,dx\right)$ of T;

```
$R := T'/T$;
$(K, S) :=$ DRCF$[i](R)$;
*return* $S(x) \exp\left(\int K(x)\,dx\right)$.
```

4.2 Additive decompositions

Definition 4 *A rational function $R \in \mathbb{F}(x)$ is said to be rational integrable if there exists an $R_1 \in \mathbb{F}(x)$ such that $R = R_1'$. Similarly, a hyperexponential function $T(x)$ over $\mathbb{F}$ is said to be hyperexponential integrable if there exists a hyperexponential function T_1 such that $T = T_1'$.*

The additive decomposition problem for hyperexponential functions can be specified as follows.

Given a hyperexponential function T, find hyperexponential functions T_1 and T_2 such that

(1) $T = T_1' + T_2$,
(2) *if T is hyperexponential integrable, then $T_2 = 0$,*
(3) *if T is not hyperexponential integrable, then T_2'/T_2 has a DRNF (K, S) such that the denominator of S has the minimal possible degree.*

This formulation agrees with that of the well-known reduction algorithms for rational functions [5, 7] since if $T_2 \in \mathbb{F}(x)$ then $\mathrm{num}(K) = 0$, $\mathrm{den}(K) = 1$, and $\mathrm{den}(S) = \mathrm{den}(T_2)$. It also includes Gosper's algorithm as a special case, i.e., $T_2(x)$ is identically zero if $T(x)$ is hyperexponential integrable.

An algorithm which solves the additive decomposition problem for hyperexponential functions and applications of this algorithm are described in [6].

5 Implementation

We have implemented the algorithms in this paper in the computer algebra system Maple. The functions are put together in the module `drnf`:

```
> print(drnf);
module()
local merge, mparfrac, simplefrac, grouping, Sgen, opgen, GosperStep3
    rightform, ReduceCert, MinimalLinearSpecialization;
export ReduceHyperexp, MultiplicativeDecomposition,
    RationalCanonicalForm, AreSimilar, Verify, IsHyperexponential,
    Zeilberger, Gosper, PolynomialNormalForm, VerifydZ;
option package;
description "a reduction algorithm for hyperexponential functions";
end module
```

The Maple source code, help pages for the exported functions, and some test samples are available from

`http://www.scg.uwaterloo.ca/~hqle/code/DRNF.html`

References

1. S.A. Abramov, H.Q. Le, M. Petkovšek. Rational canonical forms and efficient representations of hypergeometric terms. In J.R. Sendra, editor, *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, pages 7–14, 2003.
2. S.A. Abramov, M. Petkovšek. Rational normal forms and minimal decompositions of hypergeometric terms. *Journal of Symbolic Computation* **33**, No. 5, 521–543, 2002.

3. G. Almkvist, D. Zeilberger. The method of differentiating under the integral sign. *Journal of Symbolic Computation* **10**, 571–591, 1990.
4. J. Gerhard. Modular algorithms in symbolic summation and symbolic integration. Ph.D. thesis, Universität-Gesamthochschule Paderborn, 2001.
5. Ch. Hermite. Sur l'integration des fractions rationelles. *Nouvelles annales de mathematiques (2 serie)* **11**, 145–148, 1872.
6. K.O. Geddes, H.Q. Le, Z. Li. Differential rational normal forms and a reduction algorithm for hyperexponential functions. To appear in *the Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, 2004.
7. M. Ostrogradsky. De l'integration des fractions rationelles. *Bull. de la classe physico-mathématique de l'Académie Imperiale des Sciences de Saint-Petersburg* **IV** 147–168, 286–300, 1845.

A note on computing the regular solutions of linear differential systems

Sergei A. Abramov and Denis E. Khmelnov [⋆]

abramov@ccas.ru
khmelnov@ccas.ru
Russian Academy of Sciences,
Dorodnicyn Computing Centre,
Vavilova 40, 119991, Moscow GSP-1,
Russia

Summary. We present an approach to find all regular solutions of a system of linear ordinary differential equations using EG'-algorithm [2, 3] as an auxiliary tool.

1 Introduction

Let

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

where $D = d/dz$.

Assume that $Q_\rho(x), \dots, Q_0(x)$ are polynomials in z over $\mathbb{C}$. A regular solution of the equation $Ly = 0$, or, the same of the operator L, at a fixed point $z_0 \in \mathbb{C}$, is a solution of the form

$$(z - z_0)^\lambda F(z) \tag{2}$$

with $F(z) \in \mathbb{C}((z - z_0))[\log(z - z_0)]$, where $\mathbb{C}((z - z_0))$ is the field of Laurent series (here we do not consider convergence problems; all series are formal). The value λ is the *exponent* of regular solution (2). W.l.g. we will suppose that $z_0 = 0$. The problem of constructing all regular solutions of L at 0 can be solved, e.g., by Frobenius algorithm ([6]), which is based on using the indicial equation $f(\lambda) = 0$ of L at 0 (a right-hand side which contains, in particular, factors $f(\lambda)$ and z^λ, must be constructed for L; the corresponding solutions must be differentiated by λ and so on). Not only the values of roots of $f(\lambda) = 0$, each taken separately, are substantial for Frobenius' algorithm, but also multiplicities of the roots and the existence of roots differing by integers.

[⋆]The work is partially supported by the ECO-NET program of the French Foreign Affairs Ministry.

To apply Frobenius' algorithm to a system of linear difference equations one has to transform the system into a large order scalar differential equation (e.g. by the cyclic vector method). The scalar equation may have huge coefficients, that makes the approach quite unpractical.

In [4, Section 5] another algorithm for constructing regular solutions of a first order system of the form

$$y' = Ay, \quad A \in \mathrm{Mat}_N(\mathbb{C}(z)) \tag{3}$$

was described (in [5, Section 3.3] an extended version of the same algorithm was presented). This algorithm is direct, i.e., it uses neither the cyclic vector method nor any other decoupling procedure. For a given value λ the algorithm constructs a basis of regular solutions at 0 whose exponent is λ (if there exists no such solution then the basis is empty). The algorithm from [4, 5] does not need any information neither on multiplicity of λ nor on the existence of other roots with integer distance from λ. This algorithm constructs step by step a sequence of first order linear differential systems, enumerated by $0, 1, \ldots$, which are inhomogeneous starting from the system with the number 1 (the corresponding right-hand sides contain solutions of the preceding systems). If

$$z^{\lambda}\left(g_0(z) + g_1(z)\frac{\log z}{1!} + g_2(z)\frac{\log^2 z}{2!} + \cdots + g_m(z)\frac{\log^m z}{m!}\right) \tag{4}$$

is a regular solution of (3) then $g_i(z)$ is a Laurent series solution of the constructed i-th system; the process of finding of regular solutions of (3) is terminated when the current constructed system has no non-zero Laurent series solutions. It is necessary to be able to find Laurent series solutions of a given system (the recognizing of the existence included). To do this in [4, 5] a transformation of the system into a so-called *super-irreducible* form ([7]) is computed. Once the system is in a super-irreducible form then a bound of the "pole order" of the Laurent series solution, and then the coefficients of the solutions themselves can be computed directly (in turn). Additionally, if the original system is in a super-irreducible form, then one can find all possible exponents $\lambda_1, \lambda_2, \ldots$ of its regular solutions.

We describe in this paper a modification of the algorithm from [4, 5]. This modification does not use the transformation of a system into a super-irreducible form. Instead, we use EG'-algorithm from [2, 3] (see Section 3). Note that sometimes the transformation into a super-irreducible form as well as the application of EG'-algorithm is not fast. When we need to solve a linear differential system of a large size, then it could make a sense to try both approaches; if we are lucky, at least one (it is possible that only one) of them will solve the problem.

It is convenient for this purpose to reorganize the algorithm from [4, 5] in such a manner that it would be applicable to any linear system

$$Ly = 0 \tag{5}$$

where L has the form (1) with

$$Q_i(z) \in \mathrm{Mat}_N(\mathbb{C}[z]), \quad i = 0, \dots, \rho; \tag{6}$$

in particular, L can be a scalar operator of arbitrary order (in this case $N = 1$). This is done in Section 2; some useful properties of this version of the algorithm are described as well. In Section 4 the algorithm is summarized and in Section 5 we describe some computing remarks useful for implementation of the algorithm. Detailed example of the application of the algorithm is presented in Section 6. The implementation of the algorithm in Maple and related experiments are described in Section 7.

Acknowledgements. We would like to thank Prof. M. Barkatou (University of Limoges) for valuable discussions about regular solutions of linear differential systems.

2 Linear differential systems of arbitrary order

First, consider the problem of the search for Laurent series solutions of (5). We can construct the associated recurrent system $Rc = 0$ with

$$R = P_l(n)E^l + \cdots + P_t(n)E^t, \;\; P_j(n) \in \mathrm{Mat}_N(\mathbb{C}[n]), \;\; j = t, \dots, l \tag{7}$$

for the coefficients of any such solution. If $\det P_l(n)$ is the zero polynomial, then it is possible to transform the recurrent system into a system with a non-zero $\det P_l(n)$. This can be done by EG'-algorithm (see Section 3). In the rest of this section we suppose that $\varphi(n) = \det P_l(n)$, $\varphi(n) \in \mathbb{C}[n] \setminus \{0\}$.

Set $\psi(n) = \varphi(n-l)$ and n_0, n_1, resp., minimal and maximal integer roots of $\psi(n)$ (if there is no integer root, then (5) has no Laurent series solution). Any Laurent series solution of (5) has no term $c_k z^k$, $c_k \in \mathbb{C}^n$, with $k < n_0$. Using the recurrence $Rc = 0$ and the constructed constraints, we can, by a linear algebra procedure, compute a basis of the linear space of initial segments

$$c_{n_0} z^{n_0} + c_{n_0+1} z^{n_0+1} + \cdots + c_M z^M,$$

where M is a fixed integer such that $M \geq n_1$ and M is greater than all indexes involved into the constraints.

Observe, that if our differential system is inhomogeneous with a Laurent series right-hand side (the coefficients of that right-hand side are given using a recurrence), then similarly we will be able to construct a basis of the affine space of Laurent series solutions.

If $\psi(n)$ has a non-integer root λ, then the preliminary change of the depended variable $y = x^\lambda \bar{y}$ will produce a new equation $\bar{\psi}(n) = 0$, where $\bar{\psi}(n) = \psi(n - \lambda)$. Therefore we always can work with integer roots.

Apparently, the result of application of L to

$$g(z)\frac{\log^m z}{m!} \tag{8}$$

where $m \geq 0$ can be represented in the form

$$L_{m,m}(g)\frac{\log^m z}{m!} + L_{m,m-1}(g)\frac{\log^{m-1} z}{(m-1)!} + \dots + L_{m,1}(g)\frac{\log z}{1!} + L_{m,0}(g), \quad (9)$$

where the coefficients of differential operators $L_{i,j}$ belong to $\mathrm{Mat}_N(\mathbb{C}(z))$.

Proposition 1 *The coefficients of all operators $L_{i,j}$ belong to $\mathrm{Mat}_N(\mathbb{C}[z, z^{-1}])$ and, additionally,*

$$\begin{aligned} L_{0,0} &= L_{1,1} = L_{2,2} = \dots = L, \\ L_{1,0} &= L_{2,1} = L_{3,2} = \dots, \\ L_{2,0} &= L_{3,1} = L_{4,2} = \dots, \end{aligned} \quad (10)$$

....

in (9).

Proof: After the applying of L to (8) one gets, e.g., $L_{m,m-1}(g)$ by gathering together all terms that contain one time differentiated factor (8); but

$$\left(\frac{\log^m z}{m!}\right)' = \frac{1}{z} \cdot \frac{\log^{m-1} z}{(m-1)!}$$

and the new factor $1/z$ does not depend on m (due to considering $(\log^m z)/m!$ instead of $\log^m z$). ∎

Set

$$\begin{aligned} L_0 &= L_{0,0} \; (= L_{1,1} = L_{2,2} = \dots = L), \\ L_1 &= L_{1,0} \; (= L_{2,1} = L_{3,2} = \dots), \end{aligned}$$

....

If $\mathrm{ord}L = d$, then $\mathrm{ord}L_i = d - i$, $i = 0, \dots, d$, $L_{d+1} = L_{d+2} = \dots = 0$. We obtain

$$L\left(\sum_{m=0}^{k} g_{k-m}(z)\frac{\log^m z}{m!}\right) = L_0(g_0)\frac{\log^k z}{k!} + (L_1(g_0) + L_0(g_1))\frac{\log^{k-1} z}{k-1!} + \cdots$$
$$+ (L_k(g_0) + L_{k-1}(g_1) + \dots + L_0(g_k)).$$

Therefore the equality

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

is valid iff

$$\begin{aligned} L_0(g_0) &= 0, \\ L_0(g_1) &= -L_1(g_0), \end{aligned}$$

$$L_0(g_2) = -L_1(g_1) - L_2(g_0), \tag{11}$$
$$L_0(g_3) = -L_1(g_2) - L_2(g_1) - L_3(g_0),$$
$$\dots\dots\dots\dots$$

Denote S_i the system of first $i+1$ equations from (11), i.e., of the equations whose left hand sides are $L_0(g_0), \dots, L_0(g_i)$. Hence we have the following proposition.

Proposition 2 *The equality (5) has regular solution*

$$\sum_{m=0}^{k} g_{k-m}(z) \frac{\log^m z}{m!} \tag{12}$$

iff $(g_0(z), \dots, g_k(z))$ is a Laurent series solution of S_k.

Note the following. We find $g_0(z)$ using the first equation from (11). This solution may contain some arbitrary constants. When we use $g_0(z)$ in the right-hand side of the second equation from (11), some of those arbitrary constants have to be specified to make the second equation solvable in non-zero Laurent series; if this is possible, then we get $g_1(z)$ that in its turn may contain arbitrary constants and so on. So when Laurent series solutions of S_i are constructed and we solve S_{i+1}, we, in general situation, decrease the number of the arbitrary constants in solutions of S_i and find new Laurent series g_{i+1} (which may contain some arbitrary constant as well).

Denote G_k the set of all regular solutions of the form (12) of equation (5) (the case $g_0(z) = 0$ is not excluded).

Proposition 3 $G_0 \subset G_1 \subset \dots \subset G_k \subset \cdots$.

Proof: Suppose that $(g_0^*, \dots, g_i^*)$ is an $(i+1)$-tuple of Laurent series which is a solution of S_i. Set $(g_0^{**}, \dots, g_{i+1}^{**}) = (0, g_0^*, \dots, g_i^*)$. It is easy to check that $(g_0^{**}, \dots, g_{i+1}^{**})$ then satisfies the system S_{i+1}. The claimed follows from Proposition 2. ■

Apparently, if (12) is a solution of (5) $(g_0, \dots, g_k) \in \mathbb{C}((z))^{k+1}$ with $g_0 \neq 0$, then $k \leq \mathrm{ord}L - 1$. Therefore, starting from some non-negative integer k, all systems S_m, $m > k$, have only such solutions in $\mathbb{C}((z))^{k+1}$ that contain $g_0 = 0$. If k is such non-negative integer and we have constructed the set U of all solutions of S_k in $\mathbb{C}((z))^{k+1}$, then using the elements of this set we can construct all wanted regular solutions of (5). We will obtain U in the form of a vector $(g_0(z), \dots, g_k(z))$ whose entries may contain some arbitrary constants.

It is very valuable, that all equations from (11) have in the left-hand side the operator $L_0 = L$, and we have the corresponding recurrent operator for it with the non-singular leading matrix.

3 EG'-algorithm as an auxiliary tool for constructing regular solutions

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

$$P_l(n)z_{n+l} + P_{l-1}(n)z_{n+l-1} + \cdots + P_t(n)z_{n+t} = r_n \tag{13}$$

where $l \geq t$ are arbitrary integers, $z = (z^1, \ldots, z^N)^T$ is a column vector of unknown sequences (such that $z_i = (z_i^1, \ldots, z_i^N)^T$), the right-hand side r_n is a vector of polynomials in n and the matrix coefficients $P_t(n), \ldots, P_l(n)$ are polynomial in n, and $P_t(n), P_l(n)$ are non-zero. Note that it's often convenient to regard the matrix $P(n) = (P_l(n)| \ldots |P_t(n))$, which is referred to as the *explicit matrix* of the recurrence. Each of the matrices $P_t(n), \ldots, P_l(n)$ is called a *block* of the explicit matrix (resp. of the system (13)) and the matrices $P_l(n)$ and $P_t(n)$ are called *leading* and *trailing* matrices of the explicit matrix (resp. of the system (13)).

The roots of the determinants of the matrices $P_t(n)$ and $P_l(n)$ (when those matrices are non-singular over $\mathbb{C}(n)$) are always important for determining the structure of the solution space (e.g. bounds on the orders of the solutions). It may happen however that either $P_t(n)$ or $P_l(n)$ is singular. In that case, not only it is impossible to compute bounds on the orders of the solutions, but it also makes difficult, from a computational standpoint, to use the recurrence (13) to compute the sequence of vectors it generates.

A natural solution in that case is to compute an equivalence transformation of the recurrence system, which transforms it into a form with either the leading or trailing matrix nonsingular. This transformation may be a "quasi–equivalence", in the sense that the eventual changes in the solution set can be easily taken into account.

Such EG-algorithm was developed in [1] and later improved (EG'-algorithm) in [2]. It allows transforming the recurrence (13) to the form with the non-singular leading (resp. trailing) matrix. The given system is equivalent to the transformed system accompanied by a set of linear constraints (the set may be empty).

The general scheme of the algorithm is the following: if the leading (resp. trailing) matrix is singular, then we left-multiply it by another matrix (obtained for example by elimination, but not necessarily so) in order to zero one of its rows. This stage is called a *reduction* of the block. Suppose that the i-th row of the block is now zero. Then, we shift the i-th row of the transformed explicit matrix, which corresponds to left-multiplication of the i-th equation of the system (13) by the shift operator E (resp. E^{-1}) after the reduction step (so along with shifting the i-th row, we replace n by $n + 1$ (resp. $n-1$) in that row). Obviously, all the corresponding transformations are performed on the right-hand side as well. Note that the reduction step may generate a set of linear constraints because of multiplications of the transformed rows by polynomials having integer roots. Each of the constraints is a linear relation that contains a finite set of variables z_i^j. The process terminates if some special precautions are taken. To guarantee termination, it is sufficient that each reduction does not increase the width of any row: then the sum of all the widths decreases after the shift.

One of the important applications of the algorithms is solving linear functional (e.g. differential) systems with polynomial coefficients. The systems induce recurrence systems for the coefficients of their series solutions in some basis. This associated recurrence is of the form (13). EG'-algorithm is shown to be efficient enough ([3]) for the purpose and used for finding polynomial, rational, and formal series solutions of linear functional systems.

4 Algorithm

Summarizing the above information, the general scheme of finding regular solutions of the system (5) is the following:

1. For a given system S in the form (5), construct the associated matrix recurrence in the form (13). Using EG'-algorithm transform it into the recurrence of the same form but such that $\varphi(n) = \det P_l(n)$ is not identically zero. Compute all roots of $\varphi(n)$, divide them into the groups of ones having integer differences, and construct the set Λ consisting of representatives of the groups (one representative out of each group).
2. For each $\lambda \in \Lambda$ compute regular solution whose exponent is λ:
   a) Compute system S_λ by substituting $y = x^\lambda y_\lambda$. Construct the associated matrix recurrence in the form (13). Using EG'-algorithm transform it into the recurrence R_λ of the same form but such that $\varphi_\lambda(n) = \det P_l(n)$ is not identically zero. The transformed recurrence includes a set of additional constraints (the set may be empty) and the transformed right-hand side in a generic form.
   b) Determine the number M_λ of needed initial terms of Laurent series such that all integer roots of $\varphi_\lambda(n)$ and all indices of constraints are less than the number.
   c) Successively solve systems (11) for the needed number of initial terms of Laurent series using the recurrence R_λ while it's possible. It gives regular solutions y_λ of S_λ in the form (12).
3. Combine all solutions $\{y_\lambda\}_{\lambda\in\Lambda}$ into general regular solution $y = \sum_{\lambda\in\Lambda} x^\lambda y_\lambda$.

5 Computing remarks

5.1 Associated recurrence

The main part of the algorithm (see Section 4) is solving a single system from the sequence (11). As it is mentioned above all systems from (11) have in the left-hand side the same operator $L_0 = L$. Hence the associated matrix recurrences have the same left-hand side as well. But the right-hand sides of the recurrences are different. In order to regard the different right-hand sides at once during EG'-algorithm transformations we may apply all the transformations to a generic right-hand side. Then as a result of EG'-algorithm we have the transformed recurrence in the form (13) with non-singular $P_l(n)$, set of linear constraints and the transformed right-hand side in a generic form. Each component of this generic transformed right-hand side is a linear combination of possibly shifted components of the right-hand side before transformations (this is consequence of the corresponding operations in the recurrence during EG'-algorithm). In this way we can use the same transformed recurrence for solving any single system from the sequence (11) specifying the concrete right-hand side by substituting corresponding values into the generic right-hand side.

5.2 Computing the right-hand side

Computing the right-hand side is not so simple since the right-hand side for the m-th system before transformations is in the form

$$-\sum_{k=1}^{m} L_k(g_{m-k}), \tag{14}$$

where $g_0, \ldots, g_{m-1}$ are Laurent series solutions of the preceding systems in the sequence (11). Since in practice we represent Laurent series solution by segment of initial terms, we need to determine the needed numbers of initial terms of $g_0, \ldots, g_{m-1}$.

In its turn the numbers depend on the number M_λ of initial terms of transformed right-hand side which is determined on the step 2b of the algorithm (see Section 4) for all systems in the sequence (11) and ensures that next terms of the series are computed from preceding ones by simple use of the recurrence.

So we compute the transformed right-hand side in the following way:

1. Taking into account M_λ and the form of the components of transformed generic right-hand side, compute the numbers of needed initial terms of the components of right-hand side before transformations to ensure the number of initial term in the transformed right-hand side being equal to M_λ.

2. Taking into account form of the operators $L_1, \ldots, L_m$, compute the numbers of initial terms of $g_0, \ldots, g_{m-1}$ to ensure the needed numbers of initial terms of the components of right-hand side before transformations.
3. Compute the corresponding initial segments of $g_0, \ldots, g_{m-1}$.
4. Compute the initial segment of the right-hand side before transformations substituting the initial segments of $g_0, \ldots, g_{m-1}$ into (14).
5. Compute the initial segment of the transformed right-hand side substituting the initial segment of right-hand side before transformations into the transformed right-hand side in a generic form.

5.3 Extending solution component

Computing the transformed right-hand side depends on computing initial segments of $g_0, \ldots, g_{m-1}$ (step 3 in Section 5.2). Since the required number of initial terms of the solution component g_k may be greater than the number of the initial terms computed on the preceding steps of the algorithm, we need to extend the component. In order to do it we need to compute next terms using the associated recurrence. It means we need to extend the corresponding transformed right-hand side, computing it using approach from Section 5.2 substituting M_λ by new number. Note that again it may require extending other solution components. So this procedure is recursive.

5.4 Computing initial segment

When the transformed right-hand side of the recurrence is computed, solving a system from the sequence (11) for the needed number of initial terms of Laurent series may be performed one by one using the recurrence. In each step one of the following options occurs:

- the next term is computed being expressed as a function of the previous terms;
- a linear constraint on the previous terms appears, which can be either resolved or is inconsistent that means that there is no Laurent series solution;
- the next term is a new arbitrary constant (it may be defined on the next steps of the computations by resolving constraints).

After all steps either all initial terms are computed (some of them being arbitrary constants) or it's determined that there is no Laurent series solution.

Note the following:

1. Since each of the $g_0, \ldots, g_{m-1}$ may have arbitrary constants, the right-hand side for m-th system in the sequence (11) may have the same arbitrary constants. It leads to the fact that during computing the initial segment of the Laurent series solution of the m-th system some of the constants may be defined due to resolving appearing constraints. It may

lead to transforming the initial segment of g_0 to zero. Since the number of terms in the segment is determined in such a way that all the rest terms are computed in turn by the associated recurrence with the non-singular leading matrix, it gives that g_0 is identically zero. As it is noted above, if the case happens it means that all solution components are computed and m-th system has no proper Laurent series solutions.

2. As it follows from the Proposition 3 we can use only the last found solution of the form (12) as the regular solution whose exponent is λ, since it contains all previously found solutions of the form as well.

6 Example

Consider the following system:

$$\begin{aligned} &\frac{13}{2}x^2D^2y_1(x) + \frac{33}{4}xDy_1(x) + \frac{9}{8}y_1(x) + x^3D^3y_1(x) - x^2D^2y_2(x) + \\ &\qquad - 3xDy_2(x) - \frac{3}{4}y_2(x) = 0 \\ &x^2Dy_2(x) + \frac{3}{2}y_2(x) - x^2y_2(x) = 0 \end{aligned} \tag{15}$$

The explicit matrix of the associated recurrence is

$$\begin{pmatrix} (n-2)(n-1)n + \frac{13}{2}(n-1)n + \frac{33}{4}n + \frac{9}{8} & -(n-1)n - 3n - \frac{3}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & n+\frac{1}{2} & 0 & -1 \end{pmatrix},$$

with leading index of the recurrence $l = 0$, and trailing one $t = -2$. EG'-algorithm gives

$$\begin{pmatrix} (n-2)(n-1)n + \frac{13}{2}(n-1)n + \frac{33}{4}n + \frac{9}{8} & -(n-1)n - 3n - \frac{3}{4} & 0 & 0 & 0 & 0 \\ 0 & n+\frac{3}{2} & 0 & -1 & 0 & 0 \end{pmatrix},$$

with no constarints. The determinant of the leading matrix is $\frac{1}{16}(8n^3+28n^2+30n+9)(2n+3)$. The roots are $-\frac{1}{2}$ and $-\frac{3}{2}$ and they form one group. Let set of representatives $\Lambda = \{-\frac{1}{2}\}$.

After substitution $y = x^{-\frac{1}{2}}\bar{y}$, explicit matrix of the associated recurrence is

$$\begin{pmatrix} (n-2)(n-1)n + 5(n-1)n + 4n & -(n-1)n - 2n & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & n & 0 & -1 \end{pmatrix},$$

with leading index of the recurrence $l = 0$, and trailing one $t = -2$. EG'-algorithm gives

$$\begin{pmatrix} (n-2)(n-1)n + 5(n-1)n + 4n & -(n-1)n - 2n & 0 & 0 & 0 & 0 \\ 0 & n+1 & 0 & -1 & 0 & 0 \end{pmatrix}, \tag{16}$$

with no constraints. The right-hand side before transformations in a generic form is

$$\begin{pmatrix} r_n^1 \\ r_n^2 \end{pmatrix},$$

and the transformed one is

$$\begin{pmatrix} r_n^1 \\ r_{n+1}^2 \end{pmatrix} \tag{17}$$

The determinant of the leading matrix is $n(n^2+2n+1)(n+1)$. The roots are 0 and -1. Taking into account $l=0$, it means that the initial segment needs to be from x^{-1} till x^0.

Solving first system from the sequence (11) means solving the recurrence (16) with the zero right-hand side for terms from -1 till 0. It gives

$$g_0 = \begin{pmatrix} c_{0,1}x^{-1} + c_{0,2} + O(x) \\ c_{0,3}x^{-1} + c_{0,3} + O(x) \end{pmatrix}$$

Here and below $O(x^k)$ means the tail of the formal series, i.e. the terms of the power greater than or equal to k.

Compute the operator needed for right-hand side of the second system from the sequence (11):

$$L_1(y(x)) = \begin{pmatrix} y_1(x) + 7xDy_1(x) + 3x^2D^2y_1(x) - y_2(x) - 2xDy_2(x) \\ xy_2(x) \end{pmatrix} \tag{18}$$

Taking into account (17), we compute that the initial terms of the right-hand side before transformations should be up to x^1. Then from (18) we conclude that the initial terms of g_0 should be up to x^1 as well. Extending g_0 gives

$$g_0 = \begin{pmatrix} c_{0,1}x^{-1} + c_{0,2} + \frac{1}{4}c_{0,3}x + O(x^2) \\ c_{0,3}x^{-1} + c_{0,3} + \frac{1}{2}xc_{0,3} + O(x^2) \end{pmatrix}$$

That leads to the transformed right-hand side being equal to

$$\begin{pmatrix} -c_{0,3}x^{-1} - c_{0,2} + c_{0,3} - \frac{1}{2}c_{0,3}x + O(x^2) \\ -c_{0,3}x^{-1} - c_{0,3} - \frac{1}{2}c_{0,3}x + O(x^2) \end{pmatrix} \tag{19}$$

Solving the recurrence (16) with respect to the right-hand side (19) gives

$$g_1 = \begin{pmatrix} c_{1,1}x^{-1} + c_{1,2} + O(x) \\ c_{1,3}x^{-1} + c_{1,3} + O(x) \end{pmatrix}.$$

and due to resolving appearing constraints changes preceding solution component

$$g_0 = \begin{pmatrix} c_{0,1}x^{-1} + O(x^2) \\ 0 + O(x^2) \end{pmatrix}$$

Compute the next operator needed for the right-hand side of the third system from the sequence (11):

$$L_2(y(x)) = \begin{pmatrix} 2y_1(x) + 3xDy_1(x) - y_2(x) \\ 0 \end{pmatrix} \tag{20}$$

Taking into account (17), (18), and (20) we compute that the initial terms both of g_0 and of g_1 should be up to x^1. g_0 is already in the needed form and extending g_1 gives

$$g_1 = \begin{pmatrix} c_{1,1}x^{-1} + c_{1,2} + \frac{1}{4}c_{1,3}x + O(x^2) \\ c_{1,3}x^{-1} + c_{1,3} + \frac{1}{2}xc_{1,3} + O(x^2) \end{pmatrix}$$

That leads to the transformed right-hand side being equal to

$$\begin{pmatrix} -c_{1,3}x^{-1} - c_{1,2} + c_{1,3} - \frac{1}{2}c_{1,3}x + O(x^2) \\ -c_{1,3}x^{-1} - c_{1,3} - \frac{1}{2}c_{1,3}x + O(x^2) \end{pmatrix} \tag{21}$$

Solving the recurrence (16) with respect to the right-hand side (21) changes preceding component g_0 to be zero due to resolving appearing constraints. It means that all solutions components are already found and no proper g_2 exists.

Combining all the above we find the solution of (15)

$$y = \begin{pmatrix} x^{-1/2}(\ln(x) * (c_1x^{-1} + O(x^2)) + c_4x^{-1} + c_2 + \frac{1}{4}c_3x + O(x^2)) \\ x^{-1/2}(c_3x^{-1} + c_3 + \frac{1}{2}xc_3 + O(x^2)) \end{pmatrix}$$

7 Implementation and experiments

The algorithm is implemented in Maple on top of the package `LinearFunctionalSystems`, which is implementing EG'-algorithm and some algorithms for finding closed-form solutions of linear functional systems with polynomial coefficients. The algorithm is implemented as the function that returns the regular solutions of the specified linear differential system of equations with polynomial coefficients with involved Laurent series represented as their initial segments. The number of the initial terms are determined automatically to ensure that the rest terms of the series can be directly computed (in turn) using associated recurrences (i.e. the leading matrix is invertible for all the rest terms). In order to extend initial segments of the Laurent series of the found regular solution the other function is provided, which returns the regular solution with the segments extended to the specified degree.

For experiments we use as well a Maple implementation of the algorithm from [5] (presented in the package `ISOLDE`).

We compared the two programs on two types of sets of generated systems.

For the first type of the sets we generated randomly the systems of the form $Y'(x) = A(x)Y(x)$, where $A(x)$ is the matrix of rational functions, and the entries on each row of the matrix have the same denominator. For each $n \in \{4, 7, 10\}$, three sets of 20 random $n \times n$ matrices were generated. For each of the sets, numerators and denominators of the entries of the generated matrices had degrees bounded by $d \in \{4, 8, 12\}$ respectively. More precisely the following Maple instruction was used as a generator:

```
randpoly(x, terms=rand(1..floor(d/2))(), expons=rand(0..d))
```

Additionally, the probability of non-zero entries in the matrix was set to 3/5. The entire collection of matrices (as well as for the other comparisons reported here) is available at the URL `http://www.ccas.ru/~zavar/abrsa/regsol/comparisons.html`. The results for the first type of the sets are presented in Table 1, where the rows represent the degree bound on the coefficients, and the columns represent the size of the system. Each cell of the table corresponds one series of 20 systems and contains 2 fractions: the first is the number of systems solved faster by the program from `ISOLDE` over the number of systems solved faster by EG'-based one, and the second is the total CPU time (in seconds) taken by the program from `ISOLDE` for the 20 systems over the CPU time taken by EG'-based one. Additionally two numbers are indicated: the first one shows the number of solutions of the systems in the set containing logarithms and the second one shows the number of trivial (zero) solutions of the systems in the set.

Table 1. Results for the first type of the sets

	4	**7**	**10**
4	3/17 13.642/11.279 8–0	1/19 747.451/187.077 10–0	1/19 1060.768/399.171 13–0
8	2/18 17.405/10.362 4–0	2/18 276.639/312.407 9–1	0/20 627.547/164.203 13–0
12	4/16 15.609/13.251 4–2	2/18 211.671/389.345 8–2	1/19 1371.342/184.999 10–0

For the second type of the sets we constructed the systems in the following way. First for each pair l and d, where $l \in \{2, 4, 6\}$ and $d \in \{3, 5, 7\}$, we constructed 20 random scalar recurrences of the order bounded by l and coefficients of the degrees bounded by d. More precisely the following Maple instruction was used as a generator:

```
      (n-rand(-5..5)())^2*E^l+randpoly(E, terms=rand(1..l)(),
   expons=rand(l), coeffs=(()->randpoly(n, coeffs=rand(-5..5),
       terms=rand(1..floor(d/3)+1)(), expons=rand(0..d))))
```

Each scalar recurrence can be treated as being induced by scalar differential equation. So, second, for the constructed scalar recurrences we constructed corresponding scalar differential equations. Third we constructed first order differential systems corresponding to these equations. And as the last step we transformed the systems in accordance with changing function variables induced by randomly generated transformation matrices with integer entries and the probability of non-zero entries in the matrices set to 1/2 (only invertible matrices were selected). The results for the second type of the sets are presented in Table 2, where the rows represent the order bound on the source scalar recurrence, and the columns represent the degree bound on its coefficients. Cells contain the same information as in the first type, except for the numbers of logarithmic and trivial solutions since all solutions for the second type are non-trivial and logarithmic.

Table 2. Results for the second type of the sets

	3	**5**	**7**
2	0/20 9.640/4.376	1/19 22.625/25.594	5/15 64.592/185.720
4	0/20 15.811/7.390	2/18 30.248/46.553	7/13 71.404/122.832
6	0/20 21.567/8.920	1/19 45.389/23.609	9/11 125.859/517.655

As it is mentioned above, since the programs use different approaches, their weak and strong features are displayed on different systems. As we can see for the first type of the sets most systems were solved faster by the EG'-based program, however for some sets the total CPU time was less for `ISOLDE` since a few systems in these sets were solved much faster by this program. For the second type of the sets we can see both the same effect and the growth of the number of the systems solved faster by `ISOLDE` with the growth of the degree bound of the coefficients of the source scalar recurrences. So it seems like a difficult task to implement a poly-algorithm that would detect automatically the most efficient method to use for a particular input.

We conclude with a final remark: while EG'-based program has improved its efficiency after the recent update of some modules, the package `ISOLDE` has not been updated for quite a long time, so we do not exclude the possibility that further improvements in the package could lead to some changes in the table. It nevertheless reflects accurately the current status of those programs.

References

1. S.A.Abramov. *EG*-eliminations. *Journal of Difference equations and applications*, 1999, Vol. 5, 393–433.
2. S. Abramov, M. Bronstein. On solutions of linear functional systems. In *Proc. of ISSAC'2001*, ACM press, 2001, 1–6.

3. S. Abramov, M. Bronstein, D. Khmelnov. Regularization of linear recurrence systems. In *Transactions of French-Russian A.M.Lyapunov Institute, MSU*, 2003, Vol.4, 158 – 171.
4. M. Barkatou. On rational solutions of systems of linear differential equations. *J. Symbolic Computation*, 28(4 and 5), 547–568, October/November 1999.
5. M. Barkatou, E. Pfluegel. An algorithm computing the regular formal solutions of a system of linear differential equations. *J. Symbolic Computation*, 28(4 and 5), 569—587, October/November 1999.
6. E.A. Coddington, N. Levinson. Theory of ordinary differential equations. McGraw-Hill book company Inc. 1955
7. A. Hilali, W. Wazner. Formes super-irréducible de systèmes différentiels linéares. *Num. Math.* **50**, 1987, 429–449.

Newton-Hensel algorithm to compute power series solutions to non linear ODE of order 1 (abstract)

E. Hubert and N. le Roux

Evelyne.Hubert@sophia.inria.fr
INRIA Sophia Antipolis, projet CAFE
2004, route de lucioles - BP 93
F-06902 Sophia Antipolis Cedex, France
nicolas.leroux@unilim.fr
LACO, faculté des sciences
123 avenue Albert Thomas
87060 Limoges Cedex, France

A Newton method for computing the power series solution of a nonlinear ODE of order 1 satisfying regular initial conditions was introduced in [2, 3]. The basic step of the algorithm is to double the number of known coefficients of the power series solution using the linearisation of the equation at the known approximation. We generalized this method to systems of partial differential equations in [4]. The method works for so called *regular differential systems* and any polynomially nonlinear differential systems can be rewritten in terms of those [1, 5].

The Newton methods mentionned above lend themselves to modular computations. Our goal now is to recover the exact power series solution from the modular computations with a Hensel lifting. As a first stage we explore the case of first order ordinary differential equations. We first obtain a bound on the coefficients. We expect that, along the lines of the analytic results by Riquier [6], we can generalize that approach to PDE systems.

The focus of our talk is the problem of computing the power series solution $\bar{y} \in \mathbb{Q}[[t]]$ of $y' = f(t, y)$, $y(0) = y_0$ where $f \in \mathbb{Z}[t, y]$, $y_0 \in \mathbb{Z}$, up to a given order n, that is computing $\bar{y} \mod t^n$.

Note that under our hypothesis the coefficients of t^k in $\bar{y}$ can be written $\frac{\bar{y}_k}{k!}$ where $\bar{y}_k$ belongs to $\mathbb{Z}$. We show, using classical majorant technique and Cauchy inequalities, that

$$|\frac{\bar{y}_k}{k!}| \leq \frac{3}{\left(1 - \exp(\frac{-1}{2M})\right)^{n-1}} \text{ for } 0 < k < n$$

where $M = \sup\{|f(t, y+y_0)| : |t| \leq 1, |y| \leq 1\}$. This bound depends on f and n only and M can be replaced by any majorant of the supremum.

Let p be a prime number greater than n so that the denominators of the n first coefficients of $\bar{y}$ do not vanish. A power series $\tilde{y} \in \mathbb{Q}[[t]]$ is such that $\tilde{y} \equiv \bar{y} \mod (t^n, p^k)$, for some $n, k \in \mathbb{N}$ iff $\tilde{y}' \equiv f(t, \tilde{y}) \mod (t^{n-1}, p^k)$ and $\tilde{y}_0 \equiv \bar{y}_0 \mod p^k$

EITHER :

If $\tilde{y} \equiv \bar{y} \mod (t^n, p^k)$ then after the application of the operator that is obtained by Taylor formula, $\tilde{y} \equiv \bar{y} \mod (t^n, p^{2k})$.

The first step of the algorithm is to compute $\tilde{y}$ s.t. $\tilde{y} \equiv \bar{y} \mod (t^n, p)$ by Newton's method [3], but with all computations done modulo p. We then lift that modular solution by iterating the above operator N times where

$$p^{2^N} > \frac{6(n-1)!}{\left(1 - \exp(\frac{-1}{2M})\right)^{n-1}}.$$

Here again, operations are done modulo p. We then have $\tilde{y} \equiv \bar{y} \mod (t^n)$.

OR:

The first step of the algorithm is to compute $\tilde{y}$ s.t. $\tilde{y} \equiv \bar{y} \mod (t^n, p)$ by Newton's method [3], but with all computations done modulo p. We then lift that modular solution by iterating an operator that is obtained by Taylor formula. If $\tilde{y} \equiv \bar{y} \mod (t^n, p^k)$ then after the application of the operator $\tilde{y} \equiv \bar{y} \mod (t^n, p^{2k})$. We stop after N iterations, where

$$p^{2^N} > \frac{6(n-1)!}{\left(1 - \exp(\frac{-1}{2M})\right)^{n-1}}.$$

We then have $\tilde{y} \equiv \bar{y} \mod (t^n)$.

References

1. F. Boulier, D. Lazard, F. Ollivier, and M. Petitot. Computing representations for radicals of finitely generated differential ideals. Technical Report IT-306, LIFL, 1997.
2. R.P Brent and H.T Kung. Fast algorithms for manipulating formal power series. *ACM*, 25(4):581–595, october 1978.
3. K.O. Geddes. Convergence behaviour of the newton iteration for first order differential equations. In *Proceedings of EUROSAM '79*, pages 189–199, 1979.
4. Hubert and Le Roux. Computing power series solutions of a nonlinear pde system. In J.R. Sendra, editor, *ISSAC 2003*, pages 148–155. ACM Press, 2003.
5. E. Hubert. Notes on triangular sets and triangulation-decomposition algorithms II: Differential systems. In F. Winkler and U. Langer, editors, *Symbolic and Numerical Scientific Computing*, number 2630 in Lecture Notes in Computer Science, pages 40–87. Springer Verlag Heidelberg, 2003.
6. C. Riquier. *Les systèmes d'équations aux dérivées partielles.* Gauthier-Villars, Paris, 1910.

Computation of the stability boundaries in the linear hamiltonian systems with periodic coefficients

A.N.Prokopenya

prokopenya@brest.by
Brest State Technical University
Moskowskaya st. 267
224017 Brest
Belarus

Summary. We consider the hamiltonian system of linear differential equations with periodic coefficients. Using the method based on the existence of periodic solutions on the boundaries between the domains of stability and instability we have developed the algorithm for computation of the stability boundaries. The algorithm has been realized for the fourth order hamiltonian system arising in the restricted many-body problems. The stability boundaries have been found in the form of powers series, accurate to the fifth order in a small parameter. All the computations are done with the computer algebra system *Mathematica*.

1 Introduction

Let us consider the linear hamiltonian system of differential equations

$$\frac{dx}{dt} = JH(t)x, \tag{1}$$

where $x^T = (x_1, x_2, \ldots, x_{2n})$ ia a $2n$-dimensional vector whose components x_k and x_{n+k} are the canonically conjugated variables, $J = \begin{pmatrix} 0 & E_n \\ -E_n & 0 \end{pmatrix}$ and E_n is the $n \times n$ identity matrix, $H(t)$ is the real-valued $2n \times 2n$ matrix function which can be represented in the form of the converging series

$$H(t) = H_0 + \varepsilon H_1(t) + \varepsilon^2 H_2(t) + \ldots, \tag{2}$$

where ε is a small parameter. The matrix functions $H_k(t)$ $(k = 1, 2, \ldots)$ in (2) are continuous and periodic with a period T, while H_0 is a constant matrix. Besides, H_0 and $H_k(t)$ can depend on some parameters. Equations of the form (1) describe dynamical systems with intrinsic periodicity and appear in many branches of science and engineering (see, for example, [1]). Our interest to the system (1) arises because such a system occurs in studying of the stability of equilibrium solutions in the elliptic restricted many-body problems [2,3]. And we are interested in determination of the stability boundaries for the system (1) in the space of parameters.

According to the general theory of differential equations with periodic coefficients [1], the behaviour of solutions of the system (1) is determined by its characteristic exponents which are continuous functions of ε. And the system may be stable only if none of the eigenvalues of the matrix JH_0 has a positive real part. However, the system being stable for $\varepsilon = 0$ may become unstable even for very small values of $\varepsilon > 0$. So, in order to analyze the stability of system (1) we have to calculate its characteristic exponents for $\varepsilon > 0$. Using the method of a small parameter we can find them in the form of power series in ε as it was done in [2,4], for example. But if we are looking for the stability boundaries the method of infinite determinant turns out to be more effective [5,6]. The main aim of the present paper is to develop the corresponding algorithm of calculations and to realize it for the hamiltonian system of the fourth order. It should be emphasized that such calculations may be reasonably done only with a computer software. Here all the calculations are done with the computer algebra system *Mathematica* [7].

2 Properties of the characteristic multipliers

The hamiltonian systems of linear differential equations with periodic coefficients and their general properties have been studied quite well (see [1], for example). It is known that their characteristic multipliers obey the following

Rule 1. If ρ is a characteristic multiplier of the system (1) then ρ^{-1}, $\overline{\rho}$, $\overline{\rho}^{-1}$ are its characteristic multipliers as well, where $\overline{\rho}$ is a complex-conjugate value for ρ.

The Rule 1 restricts possible values of the characteristic multipliers of the hamiltonian systems. For instance, characteristic multipliers of the second order hamiltonian system ($n = 1$) may be either both real-valued and $\rho_2 = \frac{1}{\rho_1}$ or both complex-valued with unit magnitude $|\rho_1| = |\rho_2| = 1$ and $\rho_2 = \overline{\rho}_1$. In the first case the system (1) is unstable because one of its characteristic exponents has a positive real part. In the second case both characteristic exponents of the system are pure imaginary and it is stable. The cases $\rho_1 = \rho_2 = 1$ and $\rho_1 = \rho_2 = -1$ correspond to the boundary between stable and unstable behaviour and are characterized by the existence of periodic solutions of the system (1) with periods T and $2T$ respectively. Thus, we can seek the stability boundary of the second order hamiltonian system using the condition of the existence of periodic solutions. This approach was successfully realized for the Hill's equation in [8].

The fourth order hamiltonian system ($n = 2$) has four characteristic multipliers which must obey the Rule 1 as well. Hence, the system may be stable only if all its characteristic multipliers are complex-valued with unit magnitude $|\rho_1| = |\rho_2| = |\rho_3| = |\rho_4| = 1$. Geometrically these characteristic multipliers are situated on the unit circle in the complex plane, symmetrically in pairs with respect to the real axis (see Fig. 1a). If one of them leaves the circle then the rest three characteristic multipliers must leave the circle too because in the opposite case the Rule 1 will be broken. Moreover, the Rule 1 imposes restrictions on possible motion of the characteristic multipliers in the complex plane. One possibility is shown on Fig. 1b, when two characteristic multipliers being in the same semi-plane move toward each other on the circle until their coincidence and then start to move away of the circle along radial directions. It means that the system becomes unstable because magnitudes of two characteristic multipliers becomes larger that 1. If such a case is realized then the system (1) has no periodic solutions and we have to calculate its characteristic multipliers explicitly in order to find the stability boundaries. There is also another possibility when two characteristic multipliers moving on the circle toward each other coincide in the point $\rho = -1$ and then continue their motion along the real axis (see Fig 2). Similar situation can occur if two characteristic multipliers coincide in the point $\rho = 1$. In both cases the other two characteristic multipliers are situated on the unit circle, symmetrically with respect to the real axis. Again the cases $\rho = \pm 1$ correspond to the boundary between stable and unstable behaviour of the system (1), similarly as it is in the case of the second order hamiltonian system. The interesting and significant result from this analysis is that for $\rho = \pm 1$ the system (1) has periodic solutions of period T and $2T$ respectively and this property may be used for determination of the boundaries between the domains of stability and instability in the space of parameters.

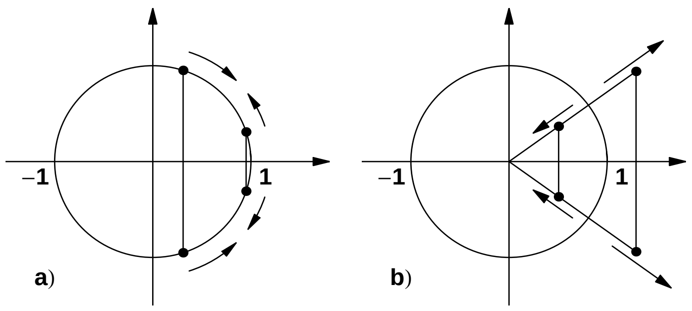


Fig. 1. Possible motion of the characteristic multipliers in the complex plane.

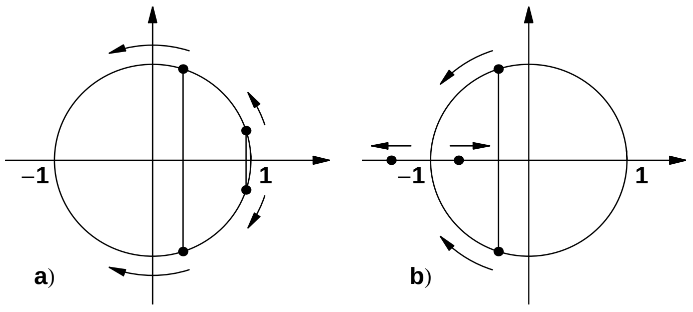


Fig. 2. Two characteristic multipliers go on the real axis.

3 Algorithm of the calculations

Let us consider the hamiltonian system (1) of the fourth order with the matrix function

$$H(t)=\begin{pmatrix}\frac{1+b+4\varepsilon\cos t}{1+\varepsilon\cos t} & 0 & 0 & -2\\ 0 & -\frac{b}{1+\varepsilon\cos t} & 0 & 0\\ 0 & 0 & 1 & 0\\ -2 & 0 & 0 & 1\end{pmatrix}, \qquad (3)$$

where b,ε are some positive parameters. Such a system arises in studying the stability of equilibrium solutions in the elliptic restricted problem of four bodies [2]. The matrix function (3) is periodic with the period $T=2\pi$ and may be represented in the form (2) where

$$H_0=\begin{pmatrix}1+b & 0 & 0 & -2\\ 0 & -b & 0 & 0\\ 0 & 0 & 1 & 0\\ -2 & 0 & 0 & 1\end{pmatrix},\quad H_k(t)=(-\cos t)^k\begin{pmatrix}-3+b & 0 & 0 & 0\\ 0 & -b & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}$$

and the corresponding series converges in the domain $|\varepsilon|<1$ for any t. The eigenvalues of the matrix JH_0 are easily found and can be represented as $\lambda_{1,2}=\pm i\sigma_1$, $\lambda_{3,4}=\pm i\sigma_2$ where

$$\sigma_{1,2}=\left(\frac{1\pm\sqrt{1-12b+4b^2}}{2}\right)^{1/2}.$$

They are distinct pure imaginary numbers if parameter b satisfy the following inequalities

$$0<b<\frac{1}{4}(6-\sqrt{32})\quad or\quad \frac{1}{4}(6+\sqrt{32})<b<3.$$

In the case we are interested in parameter b belongs to the first interval. The corresponding intervals for $\sigma_{1,2}$ are

$$\frac{1}{\sqrt{2}} < \sigma_1 < 1, \quad 0 < \sigma_2 < \frac{1}{\sqrt{2}}.$$

The analysis above shows that the domain of instability can arise only in the vicinity of the point $\sigma_2 = \frac{1}{2}$ when two characteristic multipliers $\rho_{3,4} = \exp(\pm 2\pi\sigma_2 i) = -1$. The corresponding value of b is

$$b = \frac{1}{4}(6 - \sqrt{33}). \tag{4}$$

Hence, the boundaries between the domains of stability and instability in the $b - \varepsilon$ plane are some curves $b = b(\varepsilon)$ which are characterized by the presence of periodic solutions with the period $2T = 4\pi$ and cross the b-axis in the point (4).

In order to find the stability boundaries let us rewrite the system (1) with matrix (3) in the form of two linear second order differential equations

$$\begin{cases} (1+\varepsilon\cos t)(\ddot{x}_1 - 2\dot{x}_2) + (-3+b)x_1 = 0 \\ (1+\varepsilon\cos t)(\ddot{x}_2 + 2\dot{x}_1) - b\ x_2 = 0 \end{cases} \tag{5}$$

where dot means the derivative $\frac{d}{dt}$. Now we can attempt to seek a solution of the system (5) in the form of Fourier series

$$x_1 = p_0 + \sum_{k=1}^{\infty} (p_k\ \cos(\frac{k}{2}\ t) + q_k\ \sin(\frac{k}{2}\ t)),$$

$$x_2 = \alpha_0 + \sum_{k=1}^{\infty} (\alpha_k\ \cos(\frac{k}{2}\ t) + \beta_k\ \sin(\frac{k}{2}\ t)). \tag{6}$$

On substituting (6) into equations (5) and equating coefficients of $\cos(\frac{k}{2}\ t)$ and $\sin(\frac{k}{2}\ t)$ to zero we obtain the following infinite sequence of equations

$$\begin{gathered}
(-b + \frac{13}{4} + \frac{\varepsilon}{8})\ p_1 + (1 + \frac{\varepsilon}{2})\ \beta_1 + \frac{9\varepsilon}{8}\ p_3 + \frac{3\varepsilon}{2}\ \beta_3 = 0, \\
(-1 + \frac{\varepsilon}{2})\ p_1 + (-b - \frac{1}{4} + \frac{\varepsilon}{8})\ \beta_1 - \frac{3\varepsilon}{2}\ p_3 - \frac{9\varepsilon}{8}\ \beta_3 = 0, \\
\vdots \\
\frac{\varepsilon}{8}(3-2k)^2 p_{2k-3} + \frac{\varepsilon}{2}(-3+2k)\beta_{2k-3} + (-b + \frac{13}{4} - k + k^2)\ p_{2k-1} + \\
+(-1+2k)\ \beta_{2k-1} + \frac{\varepsilon}{8}(1+2k)^2\ p_{2k+1} + \frac{\varepsilon}{2}(1+2k)\ \beta_{2k+1} = 0, \\
\frac{\varepsilon}{2}(-3+2k)p_{2k-3} + \frac{\varepsilon}{8}(-3+2k)^2\beta_{2k-3} + (1-2k)\ p_{2k-1} +
\end{gathered} \tag{7}$$

$$+(-b-\frac{1}{4}+k-k^2)\ \beta_{2k-1}+\frac{\varepsilon}{2}(1+2k)\ p_{2k+1}+\frac{\varepsilon}{8}(1+2k)^2\ \beta_{2k+1}=0,\ \ldots$$

$$\begin{gathered}
(-b+\frac{13}{4}-\frac{\varepsilon}{8})\ q_1+(-1+\frac{\varepsilon}{2})\ \alpha_1+\frac{9\varepsilon}{8}\ q_3-\frac{3\varepsilon}{2}\ \alpha_3=0,\\
(1+\frac{\varepsilon}{2})\ q_1-(b+\frac{1}{4}+\frac{\varepsilon}{8})\ \alpha_1+\frac{3\varepsilon}{2}\ q_3-\frac{9\varepsilon}{8}\ \alpha_3=0,\\
\vdots\\
\frac{\varepsilon}{8}(3-2k)^2 q_{2k-3}-\frac{\varepsilon}{2}(-3+2k)\alpha_{2k-3}+(-b+\frac{13}{4}-k+k^2)\ q_{2k-1}+\\
+(1-2k)\ \alpha_{2k-1}+\frac{\varepsilon}{8}(1+2k)^2\ q_{2k+1}-\frac{\varepsilon}{2}(1+2k)\ \alpha_{2k+1}=0,\\
\frac{\varepsilon}{2}(-3+2k)q_{2k-3}-\frac{\varepsilon}{8}(-3+2k)^2\alpha_{2k-3}+(-1+2k)\ q_{2k-1}+\\
+(-b-\frac{1}{4}+k-k^2)\ \alpha_{2k-1}+\frac{\varepsilon}{2}(1+2k)\ q_{2k+1}-\frac{\varepsilon}{8}(1+2k)^2\ \alpha_{2k+1}=0,\ \ldots
\end{gathered} \tag{8}$$

It can be seen that in fact there are two infinite subsequences of linear homogeneous equations (7)-(8). Equations (7) are for the odd coefficients p_1, p_3, ... , p_{2k-1} and β_1, ... , β_{2k-1}, while equations (8) determine coefficients q_1, q_3, ... , q_{2k-1} and α_1, ... , α_{2k-1}. Extracting coefficients of $\cos(k\ t)$ and $\sin(k\ t)$ we can easily obtain two similar subsequences of equations for the even coefficients p_{2k}, q_{2k}, α_{2k}, β_{2k}. For a solution to exist, the corresponding determinants of infinite systems (7), (8) must vanish, thus determining the stability boundaries in the $b-\varepsilon$ plane. These boundaries obviously must reduce to the point $b=(6-\sqrt{33})/4$ when $\varepsilon\to\ 0$.

Of course, it's impossible to calculate a determinant of the infinite matrix. Hence, in order to find the stability boundaries $b=b(\varepsilon)$ we should truncate the infinite subsequences of equations (7)-(8) after the k-th term, where k is a suitably large number. For $k=3$, for example, the corresponding determinants may be written as

$$D_3=\begin{vmatrix}
-b+\frac{13}{4}\pm\frac{\varepsilon}{8} & \pm1+\frac{\varepsilon}{2} & \frac{9\varepsilon}{8} & \pm\frac{3\varepsilon}{2} & 0 & 0\\
\mp1+\frac{\varepsilon}{2} & -b-\frac{1}{4}\pm\frac{\varepsilon}{8} & \mp\frac{3\varepsilon}{2} & -\frac{9\varepsilon}{8} & 0 & 0\\
\frac{\varepsilon}{8} & \pm\frac{\varepsilon}{2} & -b+\frac{21}{4} & \pm3 & \frac{25\varepsilon}{8} & \pm\frac{5\varepsilon}{2}\\
\mp\frac{\varepsilon}{2} & -\frac{\varepsilon}{8} & \mp3 & -b-\frac{9}{4} & \mp\frac{5\varepsilon}{2} & -\frac{25\varepsilon}{2}\\
0 & 0 & \frac{9\varepsilon}{8} & \pm\frac{3\varepsilon}{2} & -b+\frac{37}{4} & \pm5\\
0 & 0 & \mp\frac{3\varepsilon}{2} & -\frac{9\varepsilon}{8} & \mp5 & -b-\frac{25}{4}
\end{vmatrix}. \tag{9}$$

Equating determinants (9) to zero we obtain two algebraic equations giving an approximation for $b=b(\varepsilon)$. For sufficiently small ε we can represent the function $b=b(\varepsilon)$ in the vicinity of the point (4) as a converging power series

$$b=\frac{6-\sqrt{33}}{4}+b_1\varepsilon+b_2\varepsilon^2+\ \ldots . \tag{10}$$

Substituting (10) into (9) and equating coefficients of ε^k to zero we get the system of algebraic equations determining the coefficients b_k $(k = 1, 2, \ldots)$. Solving this system we have obtained

$$b = \frac{6-\sqrt{33}}{4} \pm \frac{\varepsilon}{8} + \frac{23\varepsilon^2}{128\sqrt{33}} \mp \frac{105\varepsilon^3}{2048} - \frac{148859\varepsilon^4}{1081344\sqrt{33}} \mp \frac{58335\varepsilon^5}{5767168} + \ \ldots, \quad (11)$$

where the error term is $O(\varepsilon^6)$. It should be emphasized that increasing the order of the determinant (9) we'll be able to find the following coefficients $b_6,\ b_7,\ \ldots$ in the expansion (10). But coefficients $b_1,\ b_2,\ \ldots,\ b_5$ found in (11) will be the same. Of course, exact expressions for the boundary $b = b(\varepsilon)$ are obtained when $k \to\ \infty$.

4 Conclusions

Using the method based on the existence of periodic solutions on the boundaries between the domains of stability and instability in the space of parameters we have developed the algorithm for computation of the stability boundaries in the hamiltonian systems of linear differential equations with periodic coefficients. The algorithm has been realized in the case of the fourth order hamiltonian system arising in the restricted many-body problems. The obtained results are in a good agreement with similar results of [2] where the calculations are done only in linear approximation on a small parameter and another method is used.

Acknowledgements
The author is very grateful to Prof. Evgeni A. Grebenikov for useful advices and fruitful discussions of the stability problem.

References

[1] V.A.Yakubovich and V.M.Starzhinskii. Linear differential equations with periodic coefficients, John Wiley, New York, 1975.

[2] E.A.Grebenikov, A.N.Prokopenya. Studying stability of the equilibrium solutions in the restricted Newton's problem of four bodies, *Bul. Academiei de Stiinte a Republicii Moldova. Matematica*, No. 2(42), 28-36 (2003).

[3] A.N.Prokopenya. Studying stability of the equilibrium solutions in the restricted many-body problems. In: *Challenging the Boundaries of Symbolic Computation - Proc. 5th International Mathematica Symposium (London, 7-11 July 2003)*, P.Mitic, Ph.Ramsden, J.Carne (Eds.), Imperial College Press, London (2003), 105-112 .

[4] A.N.Prokopenya. Calculation of the characteristic exponents for a Hill's equation. In: *Proc. 8th Rhine workshop on Computer Algebra (Mannheim, 21-22 March, 2002)*, H.Kredel, W.K.Seiler (Eds.), University of Mannheim (2002), 275-278.

[5] D.R.Merkin. Introduction to the Theory of Stability, Springer-Verlag, 1997.

[6] R.Grimshaw. Nonlinear Ordinary Differential Equations, CRC Press, 2000.

[7] S.Wolfram. The *Mathematica* book, Cambridge University Press, 1999.

[8] E.A.Grebenikov, A.N.Prokopenya. Determination of the boundaries between the domains of stability and instability for the Hill's equation, *Nonlinear Oscillations* **6**, No. 1, 42-51 (2003).

Finite presentations of spherical categories

Bruce W. Westbury

bww@maths.warwick.ac.uk
Mathematics Institute
University of Warwick
Coventry CV4 7AL

Summary. The category of representations of a group or Lie algebra has both a tensor product and a dual. These functors satisfy relations which are then taken as defining tensor categories with extra structure. Here we are concerned with pivotal and spherical categories.

Our main aim is a construction of free spherical categories. The usual approach is to use isotopy classes of planar diagrams. Our construction is purely combinatorial. The advantage of this approach is that we replace isotopy classes by a combinatorial notion of equivalence. Our construction retains the features of the diagram construction and in addition is suitable for implementation on a computer algebra system.

1 Introduction

The purpose of this paper is to develop a combinatorial theory of finitely presented spherical categories in a form amenable to implementation in a symbolic algebra package. The definition of a spherical category is given in [BW99, Definition 2.5]. Specifically a spherical category is a pivotal category in which the two different definitions of a trace map agree; and a pivotal category is a tensor category in which objects have duals, see [FY89, Definition 1.3].

The examples of spherical categories which are of interest are constructed as follows. Let V be a finite dimensional self-dual representation of a simple Lie algebra, or quantum group, or more generally of a Hopf algebra, H. Then the category of invariant tensors has objects the natural numbers and the vector spaces of homomorphisms are given by

$$\mathrm{Hom}(n,m) = \mathrm{Hom}_H(\otimes^n V, \otimes^m V)$$

This category is a pivotal category and if H is a universal enveloping algebra of a semi-simple Lie algebra or a quantum group then it is also spherical.

There is a general problem which is to take one of these examples and describe it as a finitely presented spherical category. The first author to consider this problem was Cvitanović who systematically studied all the simple Lie algebras from this point of view in [Cvi84], [Cvi77] and [Cvi]. Some examples which have been studied in detail are:

- All representations of SL(2); [CFS95] and [KL94].
- Vector representations of special linear groups, [Jon87].
- Vector representations of orthogonal and symplectic groups, [BW89].
- Fundamental representations of the rank two simple Lie algebras, [Kup96].
- Adjoint representations of SL(n), [BD02].
- Fundamental representations of SL(4), [Kim].

These papers all describe the category of invariant tensors using graphs (usually trivalent) drawn in a rectangle. This raises the problem of finding algorithms which would allow us to study these and similar categories using a symbolic algebra package. This paper proposes a solution to this problem. This approach is based on a description of oriented surfaces which first appears in [Bra21] and which is developed in [Tut79].

The original motivation for this work comes from the Deligne conjecture on the exceptional series of Lie algebras. The conjecture and supporting evidence is given in [Del96], [DdM96] and [CdM96]. The conjecture is that there is a tensor category which can be specialised to give the category of invariant tensors of the adjoint representation of any Lie algebra in the exceptional series. Our approach to this conjecture is to attempt to define this category as a finitely presented spherical category. In [Wes03a] we have shown that certain $6j$-symbols can be written in a way consistent with this conjecture. This can be taken as giving a finitely presented category which has some but not all of the properties required by the conjecture. The results of [Wes03b] can also be taken as giving relations in this category.

2 Combinatorial surfaces

The type of oriented surfaces we are considering are given by a finite amount of combinatorial data. First we consider surfaces with empty boundary.

Definition 1. *An oriented surface is the disjoint union of a finite set of oriented polygons with the edges identified in pairs by orientation reversing homeomorphisms.*

The edges of the polygons can be considered as a finite graph and for each vertex v the embedding defines a cyclic ordering of the set of edges incident at the vertex v by moving around the vertex in a clockwise direction. Conversely if we are given a finite graph together with a cyclic ordering of the set of edges incident at v for each vertex v then we can recover the oriented surface. Another description of this is to consider the set F consisting of pairs (v, e) where v is a vertex and e is an edge incident at v. These pairs have been called flags, darts, and half-edges. Then F has two operations. Imagine each edge of the graph as a rod and that we can put a bead on each rod in one of two positions, one at each end of the rod. Then F is the set of all positions for a bead. The first operation is $i : F \to F$ which moves the bead to the other end of the rod. This is an involution. The other operation $c : F \to F$ moves the bead to the next rod in a clockwise sense around the vertex. This is a bijection. Given the set F, the involution i, and the bijection c we can recover the graph and the cyclic orderings and hence the surface. The surface with the reverse orientation is given by taking (F, i, c^{-1}).

There is a dual picture where we take a pair to be a polygon and an edge of the polygon. This can be regarded as a triangle whose three vertices are the two vertices of the edge and the centre of the polygon. The action of the involution is to rotate through half a revolution about the midpoint of the edge to give the triangle on the other side of the edge. The bijection acts by rotating clockwise about the centre of the polygon.

This gives a map from connected oriented surfaces to transitive permutation representations of the free product $\mathbb{Z}_2 * \mathbb{Z}$; and so a map from connected oriented surfaces with a basepoint to finite index subgroups of the free product $\mathbb{Z}_2 * \mathbb{Z}$.

3 Cyclic graphs

3.1 Closed cyclic graphs

Here we give the definition of a cyclic graph motivated by the discussion in the introduction.

Definition 2. *A cyclic graph is a finite set F together with an involution $i : F \to F$ and a bijection $c : F \to F$.*

Definition 3. *The components of a cyclic graph F are the orbits of the group generated by i and c.*

A cyclic graph is connected if it has one component. Each component is a cyclic subgraph and F is the disjoint union of these cyclic subgraphs.

The vertices of the oriented surface are the orbits of c, the edges are the orbits of i, and the faces are the orbits of ic. Hence by counting the numbers of these orbits we can compute the genus of the oriented surface. Since we can also compute the genus of each connected component we can determine the topological type of the surface.

Definition 4. *Let Λ be a finite set with an involution $*$. Then a labelling of a cyclic graph F is a function $\lambda : F \to \Lambda$ such that $i\lambda = \lambda*$.*

Definition 5. *Let F, i, c, λ be a labelled cyclic graph. Then the labelled cyclic graph with the reverse orientation is the labelled cyclic graph $\overline{F}, \overline{i}, \overline{c}, \overline{\lambda}$ defined by*

$$\overline{F} = F \; \overline{i} = i \; \overline{c} = c^{-1} \; \overline{\lambda} = \lambda * .$$

3.2 Cyclic graphs with boundary

In this section we extend the definition of the previous section to allow cyclic graphs to have a boundary.

Definition 6. *A cyclic graph consists of a finite set F and a subset ∂F together with bijections*

$$i : F \to F \; c : F \backslash \partial F \to F \backslash \partial F \; b : \partial F \to \partial F$$

Definition 7. *For a cyclic graph F let $\sim$ be the relation on F given by $f \sim f'$ if $f' = c^n(f)$ or $f' = b^n(f)$ for some $n \in \mathbb{Z}$ or $f' = i(f)$.*

Then the relation $\sim$ is reflexive and symmetric so the transitive closure is an equivalence relation. The equivalence classes are called the components of F; and F is connected if it has a single component. Each component is a cyclic subgraph and F is the disjoint union of these cyclic subgraphs.

Definition 8. *Let Λ be a (finite) set with an involution $*$. Then a labelling of a cyclic graph F is a function $\lambda : F \to \Lambda$ such that $i\lambda = \lambda*$.*

Definition 9. *Let $F, \partial F, i, c, b, \lambda$ be a labelled cyclic graph. Then the labelled cyclic graph with the reverse orientation is the labelled cyclic graph $\overline{F}, \overline{\partial F}, \overline{i}, \overline{c}, \overline{b}, \overline{\lambda}$ defined by*

$$\overline{F} = F \; \overline{\partial F} = \partial F \; \overline{i} = i \; \overline{c} = c^{-1} \; \overline{b} = b \; \overline{\lambda} = \lambda*$$

These definitions extend the definitions of the previous section since this is the special case $\partial F = \emptyset$.

3.3 Glueing

Definition 10. *Define an equivalence relation on Λ by $\lambda_1 \sim \lambda_2$ if and only if $\lambda_1 = \lambda_2^*$. Let D be the set of equivalence classes and $\delta : \Lambda \to D$ the canonical map.*

Definition 11. *A monomial is a function $n : D \to \mathbb{N}$. The sum of two monomials is defined pointwise so that the set of monomials is the free commutative monoid on the set D.*

This means that we have indeterminates $\{\delta(\lambda) : \lambda \in \Lambda\}$ such that $\delta(\lambda_1) = \delta(\lambda_2)$ if and only if $\lambda_1 = \lambda_2^*$. Then the monomial $n : D \to \mathbb{N}$ is usually written as

$$\mu = \prod_{\delta \in D} \delta^{n(\delta)}$$

Definition 12. *A labelled boundary is a finite set X with a bijection $b : X \to X$ and a function $\lambda : X \to \Lambda$.*

The components are the orbits of the group generated by b; and a labelled boundary is connected if it has a single component. Each component is a labelled boundary and a labelled boundary is the disjoint union of its components.

Definition 13. *If X, b, λ is a labelled boundary then the labelled boundary with the reverse orientation is the labelled boundary $\overline{X}, \overline{b}, \overline{\lambda}$ defined by*

$$\overline{X} \ = X \quad \overline{b} = b^{-1} \quad \overline{\lambda} = \lambda *$$

If F is a cyclic graph then ∂F is the labelled boundary $\partial F, b|_{\partial F}, \lambda|_{\partial F}$.

Let $F, \partial F, i, c, b, \lambda$ be a labelled cyclic graph. Let B_1 and B_2 be labelled boundaries given as subsets of ∂F. Let $\theta : B_1 \to \overline{B_2}$ be an isomorphism of labelled boundaries. This means that θ is an orientation reversing isomorphism. Note that we not require B_1 and B_2 to be disjoint. Then we can glue along θ. This will give a labelled cyclic graph $F(\theta), \partial F(\theta), i(\theta), c(\theta), b(\theta), \lambda(\theta)$ and a monomial μ. The sets $F(\theta)$ and $\partial F(\theta)$ are defined by

$$F(\theta) = F \backslash (B \cup \overline{B}) \ \partial F(\theta) = \partial F \backslash (B \cup \overline{B})$$

The maps $c(\theta)$, $b(\theta)$ and $\lambda(\theta)$ are defined by

$$c(\theta) = c \ b(\theta) = b|_{\partial F(\theta)} \ \lambda(\theta) = \lambda|_{F(\theta)}$$

It remains to define $i(\theta)$ and we also associate a monomial to the glueing data.

Definition 14. *Given a labelled cyclic graph and glueing data $\theta : B \to \overline{B}$ define a relation on $B \cup \overline{B}$ by $f \sim g$ if $g = i(f)$ or $g = \theta(f)$ or $f = i(g)$ or $f = \theta(g)$.*

Then the transitive closure of this relation is an equivalence relation. Define a loop to be an equivalence class which is closed under the action of i. Each loop L is labelled by $\delta(L) \in D$. Choose $f \in L \cup \theta(L)$ and put $\delta(L) = \delta(\lambda(f))$. This is independent of the choice of f. Let $\mathfrak{L}$ be the set of loops for the glueing data and define a monomial μ by

$$\mu = \prod_{L \in \mathfrak{L}} \delta(L)$$

This is the monomial for the glueing data $\theta : B \to \overline{B}$.

Let S be an equivalence class which is not a loop. Then the set

$$\{f \in s | i(f) \notin S\}$$

contains exactly two elements say f and f'. Then define $i(\theta)(i(f)) = i(f')$ and $i(\theta)(i(f')) = i(f)$. This defines $i(\theta)(f)$ for all $f \in F$ such that $f \notin (B \cup \overline{B})$ and $i(f) \in (B \cup \overline{B})$. If $f \notin (B \cup \overline{B})$ and $i(f) \notin (B \cup \overline{B})$ then put $i(\theta)(f) = i(f)$. This defines $i(\theta)$ and the cyclic graph.

3.4 Cobordism

Then this glueing can be used to define a cobordism category. The objects of the category are labelled boundaries. Let X and Y be labelled boundaries. Then a morphism $X \to Y$ is a monomial and a cyclic graph F together with two inclusions

$$\theta_X \colon X \to \partial F \qquad \theta_Y \colon \overline{Y} \to \partial F$$

Let $F : X \to Y$ and $G : Y \to Z$ be morphisms. Then the composite is defined by first taking the disjoint union $F \cup G$. Then the inclusions $\theta_Y \colon \overline{Y} \to \partial F$ and $\theta_Y \colon Y \to \partial G$ give an isomorphism of labelled boundaries $\theta \colon (\overline{Y}_F) \to (Y_G)$ This gives glueing data for the disjoint union $F \cup G$. Let μ be the sum of the monomial for F, the monomial for G and the monomial associated to the glueing data. Then glueing along this map gives a cyclic graph $F \circ G$. Then we have

$$\partial(F \circ G) = (\partial F \backslash \theta_F(\overline{Y})) \cup (\partial G \backslash \theta_G(Y))$$

and so there is an inclusion

$$\theta_F|_X \cup \theta_G|_{\overline{Z}} : X \cup \overline{Z} \to \partial(F \circ G)$$

The composition of the two morphisms is defined to be the monomial μ and the cyclic graph $F \circ G$ with this inclusion.

The identity morphism of an object X, b_X, λ_X is the cyclic graph with

$$F = X \cup \overline{X} \ \partial F = X \cup \overline{X} \ b = b_X \cup b_{\overline{X}} \ \lambda = \lambda_X \cup \lambda_{\overline{X}}$$

Since $F \backslash \partial F$ is empty there is no definition for c. The map i is the union of the identity maps $X \to \overline{X}$ and $\overline{X} \to X$. The identity map $X \cup \overline{X} \to \partial F$ is an isomorphism of objects which makes F into a morphism $X \to X$.

Next we should check associativity. This is just the observation that two disjoint glueings can be carried out in either order.

This category has more structure. This is a strict monoidal category with tensor product given by disjoint union of cyclic graphs and addition of monomials. The unit object is the empty labelled boundary with no monomial.

3.5 Surfaces

In this section we discuss the cobordism category given by taking the realisation of the cobordism category in the previous section.

Definition 15. *A collar of order p is a cyclic graph isomorphic to the following cyclic graph.*

$$F = \{x(k), y_+(k), y_-(k), z(k) : 0 \le k < p\}$$
$$\partial F = \{x(k) : 0 \le k \le p-1\} \quad (1)$$

- *The map i is defined by $i : x(k) \longleftrightarrow z(k)$ and $i : y_+(k) \longleftrightarrow y_-(k)$ for $0 \leq k \leq p-1$.*
- *The map c is defined by $c : y_+(k) \longmapsto y_-(k)$, $c : y_+(k) \longmapsto z(k)$, $c : z(k) \longmapsto y_+(k)$*
- *The map b is defined by $b : x(k) \longmapsto x(k+1)$ where $k+1$ is taken $\pmod p$.*

Then given a cyclic graph F with boundary then there is a unique way to glue a collar to each boundary component. This gives a closed surface with a polygon for each boundary component of F. If there is no edge of the surface which occurs twice as an edge of a polygon then we can remove the interior of each of these polygons and get a surface with boundary. If an edge does appear twice as an edge of a polygon then we can still remove the interior of each of these polygons but this will give not be a surface.

3.6 Spherical categories

Next we want to consider a more general notion of glueing. This involves glueing not along whole boundary components but along intervals. The motivation is that we want to construct a spherical category whose morphisms are isotopy classes of graphs drawn in a rectangle. The composition of morphisms is given by putting one rectangle on top of the other and the tensor product is given by putting rectangles side by side.

Definition 16. *A labelled oriented interval is a finite set X and a subset ∂X partitioned into two disjoint subsets $\partial X = \partial_+ X \amalg \partial_- X$ together with inverse bijections*

$$b_+ : \partial X \backslash \partial_+ X \to \partial X \backslash \partial_- X \quad \text{and} \quad b_- : \partial X \backslash \partial_- X \to \partial X \backslash \partial_+ X$$

and a function $\lambda : X \to \Lambda$.

This implies that the finite sets $\partial_+ X$ and $\partial_- X$ have the same number of elements. A labelled interval with $\partial X = \emptyset$ is a labelled object where the bijection b is taken to be b_+.

The labelled oriented interval with reverse orientation is given by

$$\overline{X} = X \ \partial_+ \overline{X} = \partial_- X \ \partial_- \overline{X} = \partial_+ X \ \overline{b}_+ = b_- \ \overline{b}_- = b_+ \ \overline{\lambda} = \lambda^*$$

Define a relation on X by $x \sim y$ if $y = b_+(x)$ or $y = b_-(x)$. The transitive closure of this relation is an equivalence relation and the equivalence classes are the components of the interval. Each interval is the disjoint union of its components.

Definition 17. *Glueing data for an interval is a bijection θ from a subset $S \subseteq \partial_+ X$ to a subset $T \subseteq \partial_- X$ such that $\theta\lambda = \lambda*$. Given glueing data we glue along θ to get the interval Y with*

$$Y = X\backslash T \; \partial_+ Y = \partial_+ X\backslash S \; \partial_- Y = \partial_- X\backslash T$$

The maps

$$b_+ : \partial Y\backslash\partial_+ Y \to \partial Y\backslash\partial_- Y \tag{2}$$
$$b_- : \partial Y\backslash\partial_- Y \to \partial Y\backslash\partial_+ Y \tag{3}$$

are defined by

$$b_+(x) = \begin{cases} b_+(x) & \text{if} x \in \partial X\backslash\partial_+ X \\ \theta(x) & \text{if} x \in S \end{cases} \tag{4}$$
$$b_-(x) = \begin{cases} b_-(x) & \text{if} x \in \partial X\backslash\partial_- X \\ \theta^{-1}(x) & \text{if} x \in T \end{cases} \tag{5}$$

and λ_Y is just the restriction of λ_X.

Then we can define a cobordism category whose objects are labelled oriented finite sets and whose morphisms are labelled oriented intervals.

Definition 18. *An object is a finite set with a map to Λ. If A and B are objects a morphism $X\colon A \to B$ is a labelled oriented interval X together with isomorphisms of objects $A \to \partial_+ X$ and $\overline{B} \to \partial_- X$.*

Given a labelled oriented finite set S we construct the identity morphism X by taking $\partial_+ X = S$, $\partial_- X = \overline{S}$ and $X = \partial X$.

This is a tensor category under disjoint union.

Taking the reverse orientation is an anti-involution. There is a strict pivotal structure which takes a labelled oriented finite set U and constructs the labelled oriented interval $\epsilon_U = X$ with

$$X = U \amalg \overline{U}, \partial_+ \overline{X} = U, \partial_- \overline{X} = \overline{U}, b_+(u) = \overline{u}, b_-(\overline{u}) = u, \lambda = \lambda \amalg \lambda^*$$

This then gives the four morphisms ϵ_U,$\epsilon_{\overline{U}}$, $\overline{\epsilon_U}$,$\overline{\epsilon_{\overline{U}}}$. This is then a strict spherical category.

3.7 Spherical 2-categories

Labelled oriented intervals are the 1-morphisms of a strict 2-category. This strict 2-category is modelled on the strict 2-category of surfaces with corners. This 2-category is studied in [Law93],[BD95]. A closely related 2-category of 2-tangles is considered in [BL03].

Before constructing this 2-category we need to generalise the definition of glueing data and glueing given in §3.3. This generalisation consists in replacing the labelled boundaries B_1 and B_2 by labelled oriented intervals.

The 1-morphisms of this 2-category are the labelled oriented intervals. Let $X, Y\colon A \to B$ be two 1-morphisms. Then a 2-morphism $\Sigma\colon X \to Y$ is given by a monomial together with the following data. First note that the two inclusions

$$A \cup \overline{B} \to X \qquad \overline{A} \cup B \to \overline{Y}$$

gives glueing data on the disjoint union $X \cup \overline{Y}$. If we then glue along this data we get a labelled boundary $(X \cup_{A \cup B} Y)$. Then a 2-morphism $\Sigma\colon X \to Y$ is a cyclic graph Σ together with an identification of the boundary, $\partial\Sigma$, with $(X \cup_{A \cup B} Y)$.

Then to define the 2-category we are required to define a horizontal and vertical composition of 2-morphisms. The vertical composition is defined essentially as follows. Let $\Sigma_1\colon X \to Y$ and $\Sigma_2\colon Y \to Z$ be 2-morphisms. Then the two inclusions of Y in $\Sigma_1 \cup \Sigma_2$ gives glueing data. Glueing along this data gives the composite $\Sigma_1 \cup_Y \Sigma_2\colon X \to Y$.

The horizontal composition is defined essentially as follows. Let $X_1, Y_1\colon A \to B$ and $X_2, Y_2\colon B \to C$ be 1-morphisms and let $\Sigma_1\colon X_1 \to Y_1$ and $\Sigma_2\colon X_2 \to Y_2$ be 2-morphisms. Then we compose the 1-morphisms to give

$$(X_1 \cup_B X_2), (Y_1 \cup_B Y_2)\colon A \to C$$

and then the horizontal composite is a 2-morphism

$$\Sigma_1 \cup_B \Sigma_2\colon (X_1 \cup_B X_2) \to (Y_1 \cup_B Y_2)$$

In both cases to define the composition we also need to take into account the monomials and loops. In each case we define the monomial for the composite to be the sum of the monomials for the two initial 2-morphisms with the monomial given by taking into account any loops that arise from the glueing.

Then this defines a strict 2-category. We omit the proof of this claim. The conditions that need to be checked are of two types. One type consists in the constructions of identity morphisms and the conditions associated with these. The other type are associative and compatibility conditions for the two compositions. The check that these conditions are satisfied amounts to checking that two disjoint glueings can be carried out in either order.

This 2-category has additional structure. First it has a monoidal structure given by the tensor product. Furthermore it is a monoidal 2-category with duals (as defined in [BL03]).

At present there is no definition of a spherical 2-category but we suggest that this example satisfies some further conditions which should be taken as the definition.

4 Presentations

First we discuss finitely generated free spherical categories.

Definition 19. *A star of valence k is a cylic graph isomorphic to the following cyclic graph*

$$F = \{x(k), y(k) \ : 0 \leq k \leq p-1\} \qquad \partial F = \{x(k) : 0 \leq k \leq p-1\}$$

- *The map i is defined by $i : x(k) \longleftrightarrow y(k)$ for $0 \leq k \leq p-1$.*
- *The map c is defined by $c : y(k) \mapsto y(k+1)$ where $k+1$ is taken* $\pmod p$.
- *The map b is defined by $b : x(k) \mapsto x(k+1)$ where $k+1$ is taken* $\pmod p$.

If F is a cyclic graph then a vertex is an orbit of the bijection $c : F\backslash\partial F \to F\backslash\partial F$. The valency is the order of the orbit. Each vertex v of valency k gives a star S of valency k by taking $S = v \cup i(v)$ and $\partial S = i(v)$.

Let $\mathfrak{S}$ be a finite set of labelled stars. Then there is a category for which a morphism is a morphism as above together a function from the vertices of the cyclic graph to $\mathfrak{G}$ and an isomorphism of labelled cyclic graphs between the star at the vertex and the corresponding element of $\mathfrak{S}$. Then these are the morphisms of a strict spherical category $\mathcal{S}(\mathfrak{S})$. This category will be called the free strict spherical category generated by $\mathfrak{S}$. The justification for this is that it has the following universal property.

Theorem 1. *Let $\mathcal{S}$ be any strict spherical category and let $\Phi : \mathfrak{S} \to \mathcal{S}$ be any set map. Then there is a unique strict spherical functor $\mathcal{S}(\mathfrak{S}) \to \mathcal{S}$ which extends Φ.*

The uniqueness is clear. In order to see this take the generators and for each object the four morphisms given by the pivotal structure. Then we have to show any morphism can be built from these using composition and tensor product. Draw the graph in the rectangle such that any two critical points have different heights. Here a critical point is either a maximum a minimum or a vertex. Then reading the critical points in decreasing height gives an expression for the diagram of the required form.

The main issue is to show that the functor is well-defined. This means that any two such expressions give the same morphism. This uses the identities of a spherical category. This can be proved using [FY92, Theorem 5.3] to construct the free pivotal category. Denote this free pivotal category by $\mathcal{P}(\mathfrak{S})$. Then by the universal property of this category we have unique pivotal functors $\mathcal{P}(\mathfrak{S}) \to \mathcal{S}(\mathfrak{S})$ and $\mathcal{P}(\mathfrak{S}) \to \mathcal{S}$. Then the theorem follows by observing that there is a unique spherical functor $\mathcal{S}(\mathfrak{S}) \to \mathcal{S}$ such that the functor $\mathcal{P}(\mathfrak{S}) \to \mathcal{S}$ is the composite

$$\mathcal{P}(\mathfrak{S}) \to \mathcal{S}(\mathfrak{S}) \to \mathcal{S}$$

As an illustration of the difference between these; the free pivotal category on a single self-dual generator is given in [FY89, Theorem 4.1.1] as the category of crossing-free tangles whereas the construction we consider gives the diagram category given in [FY89, Definition 4.1.2].

The examples which motivated this work are slightly different. In these examples the set $\mathfrak{S}$ of generators has the property that two elements are isomorphic if and only if they have isomorphic boundary. Then we define a morphism to be a morphism as above such that for vertex there exists an isomorphism of labelled cyclic graphs between the star at the vertex and some element of $\mathfrak{S}$. The distinction is that in the first instance we had to specify the isomorphism and this isomorphism was part of the data. In the second instance the requirement is just that an isomorphism exists; if there are several such isomorphisms then we are not required to specify any one in particular.

There is a common generalisation of both of these situations. For each element of $\mathcal{S}$ we also specify a subgroup of the automorphism group of each element. In the free case this subgroup is trivial and in our examples it is the automorphism group. Now we read two morphisms as the same if there are related by a specified automorphism at each vertex. The universal property is now

Theorem 2. *Let $\mathcal{S}$ be any strict spherical category and let $\Phi\colon \mathfrak{S} \to \mathcal{S}$ be any set map which respects the specified symmetries. Then there is a unique strict spherical functor $\mathcal{S}(\mathfrak{S}) \to \mathcal{S}$ which extends Φ.*

4.1 Relations

Now that we have the notion of a free spherical category we introduce the notion of a finitely presented linear spherical category. Let K be a commutative ring and form the free K-linear category on the free spherical category $\mathcal{S}(\mathfrak{S})$. Denote this strict spherical K-linear category by $K\mathcal{S}(\mathfrak{S})$. Now define a set of relations to be a set, $\mathfrak{R}$, of morphisms of this category. Then take the tensor ideal generated by $\mathfrak{R}$ and denote it by $\langle\mathfrak{R}\rangle$; this is the intersection of all tensor ideals which contain $\mathfrak{R}$. Then the quotient category $K\mathcal{S}(\mathfrak{S})/\langle\mathfrak{R}\rangle$ is a strict spherical category with the following universal property. This is called a finite presentation if $\mathfrak{S}$ and $\mathfrak{R}$ are both finite sets.

Let $\mathcal{S}$ be any strict spherical K-linear category and let $\Phi : \mathfrak{S} \to \mathcal{S}$ be any set map. Extend this to a strict spherical K-linear functor $\Phi : K\mathcal{S}(\mathfrak{S}) \to \mathcal{S}$. If $\Phi(r) = 0$ for all $r \in \mathfrak{R}$ then Φ factors through a strict spherical K-linear functor $K\mathcal{S}(\mathfrak{S})/\langle\mathfrak{R}\rangle \to \mathcal{S}$.

As is usual when working with finite presentations the most useful presentations are the confluent presentations.

Instead of working with the Hom-sets we fix a closed boundary B and consider the set of connected cyclic graphs, F,with an isomorphism $B \to \partial F$ and where the cyclic graph has a specified genus. Call this set $V(B, g)$.

Definition 20. *A reduction order is a partial order on $V(B, g)$ with no infinite descending sequence and which is compatible with glueing. This means that if we are given the same glueing data θ on f and on g and the result of glueing is f_θ and g_θ then $f_\theta < g_\theta$.*

Definition 21. *Given a finite linear combination of labelled oriented surfaces*

$$\underline{R} = \sum_{x \in X} k_x r_x$$

the support is the subset $\operatorname{Supp}(\underline{R}) \subset X$ *defined by*

$$\mathrm{Supp}(\underline{R}) = \{x \in X | k_x \neq 0\}$$

Definition 22. *A disc is a labelled oriented surface which is connected and which has Euler characteristic one.*

This is equivalent to saying that the geometric realisation has one boundary component and is contractible.

Definition 23. *A rewrite rule, W, has a left hand side, L, and a right hand side, $\underline{R}$. The left hand side is a labelled oriented disc with boundary. The right hand side is a finite linear combination of labelled oriented discs with boundary,*

$$\underline{R} = \sum_{x \in X} k_x R_x$$

where $k_x \in K$ and R_x is a labelled oriented surface with an isomorphism of objects $\partial R_x \to L$ for each $x \in X$. Furthermore $R_x < L$ for all $x \in \mathrm{Supp}(\underline{R})$.

Then a reduction is an application of a rewrite rule. This means if a labelled oriented surface S can be written as $S = S' \cup L$ where L is a left hand side then we have glueing data which gives the finite linear combination of labelled oriented surfaces

$$\sum_{x \in S} k_x S' \cup R_x$$

This is abbreviated to $S \to S' \cup \underline{R}$. Then we can define a reduction of a finite linear combination of labelled oriented surfaces to be a reduction of one of these labelled oriented surfaces.

$$\sum_{x \in X} k_x S_x \to k_x \left(S' \cup \underline{R}\right) + \sum_{y \neq x} k_y S_y$$

This is abbreviated to $\underline{S} \to \underline{R}$.

There is no difficulty in giving an algorithm for finding all possible reductions of a given surface. This is one of the main motivations for developing these categories this way.

Then we introduce the transitive closure of the relation $\to$. This means that $\underline{S} \searrow \underline{R}$ if there is a finite sequence $\underline{S} = \underline{S_0 S_1} \dots \underline{S_n} = \underline{S}$ such that $\underline{S_{i-1}} \to \underline{S_i}$ for $1 \leq i \leq n$. We also introduce the equivalence relation generated by the relation $\to$. This means that $\underline{S} \leftrightarrow \underline{R}$ if there is a finite sequence $\underline{S} = \underline{S_0 S_1} \dots \underline{S_n} = \underline{S}$ such that $\underline{S_{i-1}} \to \underline{S_i}$ or $\underline{S_{i-1}} \leftarrow \underline{S_i}$ for $1 \leq i \leq n$.

Once we have this set-up we can define confluence, local confluence and the Church-Rosser properties and show they are all equivalent by following [Sim94, Chapter 2]. Therefore we can simply refer to a set of generators, reduction ordering and relations as confluent.

The point of local confluence is that if we are given a finite presentation then there is an algorithm for deciding if the presentation is confluent. This just involves finding overlaps and checking local confluence for each overlap.

This is a finite calculation as there can only be finitely many overlaps; and there is an algorithm for finding all these overlaps.

If a presentation is not confluent then we can attempt to complete it to a confluent presentation using the Knuth-Bendix algorithm. This procedure may not be well-defined if the reduction order is a partial order and not a total order.

References

[BD95] John C. Baez and James Dolan. Higher-dimensional algebra and topological quantum field theory. *J. Math. Phys.*, 36(11):6073–6105, 1995.

[BD02] Georgia Benkart and Stephen Doty. Derangements and tensor powers of adjoint modules for $\mathfrak{sl}_n$. *J. Algebraic Combin.*, 16(1):31–42, 2002.

[BL03] John C. Baez and Laurel Langford. Higher-dimensional algebra. IV. 2-tangles. *Adv. Math.*, 180(2):705–764, 2003.

[BR01] Béla Bollobás and Oliver Riordan. A polynomial invariant of graphs on orientable surfaces. *Proc. London Math. Soc. (3)*, 83(3):513–531, 2001.

[BR02] Béla Bollobás and Oliver Riordan. A polynomial of graphs on surfaces. *Math. Ann.*, 323(1):81–96, 2002.

[Bra21] H. R. Brahana. Systems of circuits on two-dimensional manifolds. *Ann. of Math. (2)*, 23(2):144–168, 1921.

[BW89] Joan S. Birman and Hans Wenzl. Braids, link polynomials and a new algebra. *Trans. Amer. Math. Soc.*, 313(1):249–273, 1989.

[BW99] John W. Barrett and Bruce W. Westbury. Spherical categories. *Adv. Math.*, 143(2):357–375, 1999.

[CdM96] Arjeh M. Cohen and Ronald de Man. Computational evidence for Deligne's conjecture regarding exceptional Lie groups. *C. R. Acad. Sci. Paris Sér. I Math.*, 322(5):427–432, 1996.

[CFS95] J. Scott Carter, Daniel E. Flath, and Masahico Saito. *The classical and quantum 6j-symbols*, volume 43 of *Mathematical Notes*. Princeton University Press, Princeton, NJ, 1995.

[Cvi] Predrag Cvitanović. Group theory. http://www.nbi.dk/GroupTheory/.

[Cvi77] Predrag Cvitanović. Group theory. preprint, Oxford University, 1977.

[Cvi84] Predrag Cvitanović. *Group Theory*. Nordita, 1984.

[DdM96] Pierre Deligne and Ronald de Man. La série exceptionnelle de groupes de Lie. II. *C. R. Acad. Sci. Paris Sér. I Math.*, 323(6):577–582, 1996.

[Del96] Pierre Deligne. La série exceptionnelle de groupes de Lie. *C. R. Acad. Sci. Paris Sér. I Math.*, 322(4):321–326, 1996.

[FY89] Peter J. Freyd and David N. Yetter. Braided compact closed categories with applications to low-dimensional topology. *Adv. Math.*, 77(2):156–182, 1989.

[FY92] Peter Freyd and David N. Yetter. Coherence theorems via knot theory. *J. Pure Appl. Algebra*, 78(1):49–76, 1992.

[Jon87] V. F. R. Jones. Hecke algebra representations of braid groups and link polynomials. *Ann. of Math. (2)*, 126(2):335–388, 1987.

[JS91] André Joyal and Ross Street. The geometry of tensor calculus. I. *Adv. Math.*, 88(1):55–112, 1991.

[Kim] Dongseok Kim. Graphical calculus on representations of quantum lie algebras.

[KL94] Louis H. Kauffman and Sóstenes L. Lins. *Temperley-Lieb recoupling theory and invariants of* 3*-manifolds*, volume 134 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ, 1994.

[KL01] Thomas Kerler and Volodymyr V. Lyubashenko. *Non-semisimple topological quantum field theories for 3-manifolds with corners*, volume 1765 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2001.

[Kup94] Greg Kuperberg. The quantum G_2 link invariant. *Internat. J. Math.*, 5(1):61–85, 1994.

[Kup96] Greg Kuperberg. Spiders for rank 2 Lie algebras. *Comm. Math. Phys.*, 180(1):109–151, 1996.

[Law93] R. J. Lawrence. Triangulations, categories and extended topological field theories. In *Quantum topology*, volume 3 of *Ser. Knots Everything*, pages 191–208. World Sci. Publishing, River Edge, NJ, 1993.

[Sim94] Charles C. Sims. *Computation with finitely presented groups*, volume 48 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1994.

[Tut79] W. T. Tutte. Combinatorial oriented maps. *Canad. J. Math.*, 31(5):986–1004, 1979.

[Wes03a] B. W. Westbury. Invariant tensors and diagrams. *Internat. J. Modern Phys. A*, 18(Supplement, October):49–82, 2003.

[Wes03b] Bruce W. Westbury. R-matrices and the magic square. *J. Phys. A*, 36(7):1947–1959, 2003.

[Yet01] David N. Yetter. *Functorial knot theory*, volume 26 of *Series on Knots and Everything*. World Scientific Publishing Co. Inc., River Edge, NJ, 2001.

Galois Cohomology: Some aspects of computation and applications (abstract)

Sergei Haller

Sergei.Haller@math.uni-giessen.de
Mathematisches Institut
Justus-Liebig-Universität Giessen
Arndtstr. 2, Giessen
Germany

This is a joint project with Arjeh M. Cohen, Scott H. Murray and D. E. Taylor. We design and implement algorithms for computation with groups of Lie type in the software package Magma. The goal is to perform computations with parametrised group elements.

Algorithms for computing with elements of untwisted groups of Lie type are known and implemented in Magma [2]. (Cohen, Murray, Taylor [1]; Haller [3]; Riebeek [4])

The twisted groups of Lie type are fixed point subgroups in untwisted groups of Lie type. The possible twists for a given field extension and group type are classified by Galois cohomology. We report on current work to make Galois cohomology effective.

Let G be a simple linear algebraic group defined over the field k. One step in the process of computing the Galois cohomology of G is to extend 1-cocycles on a factor group A/B to 1-cocycles on A, where A is the group of algebraic automorphisms of G and B the connected component of A. We will discuss an algorithm to solve this problem.

Galois cohomology can also be used to compute all maximal tori of G. For finite k, the tori are computed as subgroups $T_{\gamma w}$ of $G(K)_{\gamma w}$, where γ is the generator of $\mathrm{Gal}(K : k)$ and w is conjugation by an element normalising the split maximal torus T of $G(K)$. Using Lang's theorem the tori can be conjugated from $G(K)_{\gamma w}$ back into the original group $G(k)$.

References

1. Arjeh M. Cohen, Scott H. Murray, and D. E. Taylor, *Computing in groups of Lie type*, Math. Comp. (to appear), 1–22, electronically published on July 7, 2003.

2. W. W. Bosma and J. J. Cannon, *Handbook of magma functions*, School of Mathematics and Statistics, University of Sydney, Sydney, 1997.
3. Sergei Haller, *Entwicklung und Implementierung eines Algorithmuses zur Berechnung von Kommutatoren unipotenter Elemente in Chevalley-Gruppen*, Diplomarbeit, Universität Gießen, 2000.
4. R. J. Riebeek, *Computations in association schemes*, Ph.D. thesis, Eindhoven University of Technology, 1998.

Computing in groups of Lie type (abstract)

Scott H. Murray

University of Sydney
Sydney, Australia/
Techinsche Universiteit Eindhoven
Eindhoven, The Netherlands

The groups of Lie type are among the most important structures in modern mathematics. Examples of such groups include reductive Lie groups, reductive algebraic groups, and finite groups of Lie type (which include most of the finite simple groups). Many problems in the representation theory of groups of Lie type have been solved using computers (for example, by the CHEVIE group). In this talk, I discuss descriptions of these groups (via the Steinberg presentation or highest weight representations) that are useful for computation. I also give methods for dealing with the twisted groups using Galois cohomology and Lang's theorem.

This talk describes joint work with Arjeh Cohen, Sergei Haller, and Don Taylor.

Presentation of CoCoA 5

J Abbott, A Bigatti, M Caboara, M Kreuzer, and L Robbiano

Dipartimento di Matematica
Via Dodecaneso 35
Genova 16146
Italy

1 Short Abstract

One of the products of the long running CoCoA project specialised in Computations in Commutative Algebra is a freely available interactive system offering good implementations of many algorithms in that area. This program (currently CoCoA 4.3) has been, and still is, highly successful; indeed it has grown far beyond what was foreseen when its foundations were laid. The similarly specialized systems Macaulay 2 and Singular have their own special strengths.

CoCoA 5 is an important new phase in the project: the program is being completely rewritten in C++. This obviates various limitations innate in the earlier design (written in C). CoCoA 5 will be available in three guises: an interactive system, a C++ library, and a server (using an OpenMath-like interface). Ease of use is a high priority for all three forms. Currently there is an "alpha" release of the library.

The new library already offers facilities for computing Gröbner bases beyond those present in version 4.3: *e.g.* more coefficient types, and better execution speed. Version 4.3 also boasts a range of other skills including polynomial factorization, exact linear algebra, Hilbert functions, and computing with zero-dimensional schemes. CoCoA 5 will gradually grow to cover each of these skills, and ultimately extend the repertoire.

Unlike general purpose symbolic computation systems CoCoA does not offer facilities for calculus or any other area not closely related to commutative polynomial algebra. The CoCoA project is closely allied to this book:

M. Kreuzer, L. Robbiano *Computational Commutative Algebra 1*, Springer (2000), ISBN 3-540-67733-X

2 Long Abstract

The CoCoA project began in 1987. Initially it comprised just a small program in Pascal running only on Macintoshes, and written simply to enable the authors to experiment with Buchberger's algorithm for computing Gröbner bases. Within two years it had become widely used in many countries both for research and teaching. Later the program was translated into C, and ported to other platforms. As its range of abilities grew continually so did its use by researchers and teachers.

The most recent version, CoCoA 4.3, benefits from development directed at areas too often neglected in academic environments notwithstanding their unquestionably vital contributions to the usefulness of the program: *e.g.* a sophisticated (graphical) user interface, comprehensive documentation, and robustness. Naturally, there has also been continual development in more academically "respectable" areas, often spurred on by symbiosis with researchers using CoCoA not only in algebraic geometry but also in other fields such as mathematical analysis, and statistics.

An important purpose of the CoCoA program is to provide a "laboratory" for studying computational commutative algebra: it together with Singular and Macaulay 2 form an elite group of highly specialized systems having as their main forte the capability to calculate Gröbner bases. Although a number of general purpose symbolic computation systems (*e.g.* REDUCE and Maple) do offer the possibility to compute Gröbner bases, their non-specialist nature implies a number of severe compromises which make them far less suitable to act as a laboratory: *e.g.* relatively poor execution speed and limited control over the algorithm parameters.

Aside from computing Gröbner bases CoCoA's particular strengths include polynomial factorization, exact linear algebra, computing Hilbert functions, and computing with zero-dimensional schemes. The usefulness of these technical skills is enhanced by the mathematically natural language for describing computations. This language is readily learned by students, and enables researchers to explore and develop new algorithms without the administrative tedium necessary when using "low-level" languages.

An extensive evolution from humble beginnings has led to some deeply rooted limitations which cannot easily be eradicated: *e.g.* restricted coefficient types, inaccessibility as a library, and poor interaction with other symbolic computation software. A completely new incarnation, CoCoA 5, addresses all these issues; indeed, significant progress has already been made in respect of each weakness. A crucial design decision was the passage from C to C++ as the implementation language: the improved expressivity of C++ allows the source code to be more readable while offering better run-time performance. We expect concomitant benefits for future maintenance of the source. The new design is expressly as a library; a server (communicating via OpenMath) and a standalone interactive system will be built on top of this library. The new interactive interface will not gratuitously violate backward compatibility though some incompatible changes are expected.

The main challenge in the design of CoCoA 5 was to reconcile two traditionally conflicting goals: flexibility and efficiency. The inheritance mechanism of C++ plays a vital role here. Our use of inheritance is exemplified by the way in which rings and their elements are implemented. An early design decision was to allow arbitrary (commutative) rings wherever possible. The run-time penalty for this additional abstraction is virtually negligible.

Without doubt, CoCoA's primary remit is to compute with polynomials in a commutative setting, yet there are a number of interesting areas of "almost commutative" algebra where the Gröbner basis methods offered in CoCoA may practicably be applied. The library includes a prototype implementation of Weyl Algebras, and further work in this area is anticipated.

Undoubtedly the principal element of CoCoA 5 is its code for computing Gröbner bases. Theoretically the issue of computing Gröbner bases was resolved by Buchberger's algorithm, published almost forty years ago. In practice, there is often a huge gulf between the neat elegance of a published algorithm, and the complex engineering hidden within a refined implementation. In the case of Buchberger's algorithm this gulf is especially wide: as witness we cite the numerous research papers published in the interim, and the fact that studies are still being actively pursued.

Creating the new library from scratch makes it easy to include the best of these recent developments, and the carefully structured design should accommodate future ideas readily. Some of the modern ideas which are implemented include geobuckets, binary divisibility mask, fraction free representation, sophisticated reducer selection strategy, automatic homogenization, additional reducers, and multiple-float coefficients. Our use of C++ inheritance offers a clean mechanism for handling special cases: *e.g.* computing Gröbner bases of binomial and toric ideals.

Symbolic Manipulation with FORM (abstract)

Jos Vermaseren

NIKHEF, Amsterdam
The Netherlands

FORM is a symbolic system for the fast manipulation of large expressions. I will show a number of its features and properties.

The quest for correctness

Henk Barendregt

NIII, Department of Computer Science,
University of Nijmegen, Nijmegen
The Netherlands

Automated Proofs using Bracket Algebra with *Cinderella* and *OpenMath*

Dan Roozemond

Eindhoven University of Technology
Department of Mathematics and Computer Science

Summary. This paper describes the results of a project intended to make it possible to put forward geometrical theorems by pointing and clicking, and then obtain a proof for that theorem automatically. This goal was achieved by adding various options to *Cinderella* [1], a computer program with which one can create geometrical configurations. Its internal 'Randomized prover' is able to discover theorems automatically.

In the project the functionality was added to find proofs for these theorems with the aid of the computer algebra package *GAP* [9]. Communication between these two programs and the various steps in generating the proof is done by means of *OpenMath* [5, 7]. The proofs are represented by bracket calculations as proposed in [8].

1 Introduction

Proof is the idol before whom the pure
mathematician tortures himself.
Sir Arthur Eddington (1882 - 1944)

Cinderella is a computer program, with which one can create geometrical configurations. Cinderella has a built in 'Randomized prover', that is able to discover geometrical theorems and return a probabilistic proof [2]. However, these proofs are not verifiable.

The main goal of the project covered here was to make it possible to put forward a theorem by pointing and clicking (in Cinderella), and then obtain a mathematically sound proof of that theorem. Moreover, this proof should be verifiable. This goal was achieved by using the following three packages:

Cinderella "Software for doing geometry on the computer, designed to be both mathematically robust and easy to use" [1].

OpenMath "A new, extensible standard for representing the semantics of mathematical objects" [5] - The communication between Cinderella and

GAP was implemented with OpenMath. This was done using the Riaca OpenMath library for Java [7].

GAP "GAP – Groups, Algorithms, and Programming" [9].

In this paper we first introduce the OpenMath standard in Section 2. Sections 3 and 4 explain how bracket calculations are used for the representation of geometrical configurations and theorems. Section 5 addresses the structure of the prover. Section 6 gives some notes on the translation from a geometric configuration in Cinderella into bracket equations as well as the implementation using GAP. A few examples of the theorem prover in action can be found in Section 7.

2 The OpenMath Standard

The OpenMath standard is made for the representation of mathematics in such a way that mathematical objects can easily be exchanged between computer programs.

> OpenMath is an emerging standard for representing mathematical objects with their semantics, allowing them to be exchanged between computer programs, stored in databases, or published on the worldwide web. While the original designers were mainly developers of computer algebra systems, it is now attracting interest from other areas of scientific computation and from many publishers of electronic documents with a significant mathematical content. [5, Overview]

A rough overview of the standard can be found in Figure 1. The 3 layers are explained as follows:

Language The OpenMath language defines the 'grammar'. It defines notions like Variables, Constants, Errors, and Functions.

Content Dictionary A Content Dictionary (CD) is (or can be) defined for each area of Mathematics. For example the 'arith1' CD describes the notions of 'minus', 'plus', 'power', etc.

Phrasebooks A Phrasebook provides communication between OpenMath and another program. Phrasebooks exist for, for example, Mathematica, GAP and Singular. A specific Phrasebook consists of three parts:

- An *encoder* to encode OpenMath objects into commands that the program understands,
- A *decoder* to translate program output into OpenMath objects,
- The physical communication between the program and the Java (or C, or C++) program containing the OpenMath objects.

The interested reader is encouraged to have a look at `http://www.openmath.org` for an extensive overview of the OpenMath standard. In this project the (experimental) `plangeo` codec [6] is used, since it contains elements for representing planar geometry.

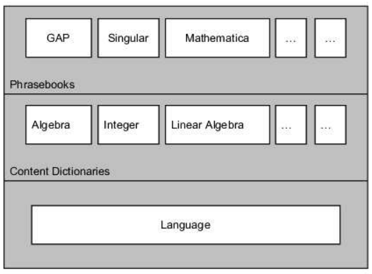


Fig. 1. The OpenMath framework

3 Brackets and Projective Geometry

When we restrict ourselves to geometric theorems that are invariant under projective transformations, we can prove geometric theorems much faster than when we would have used Gröbner bases. The method we use is based on a paper by Jürgen Richter-Gebert in 1995 [8]. For now, we only consider configurations and theses of the form:

- The three points A, B, and C lie on one line (the points A, B, and C are collinear), denoted by '$h(A,B,C)$',
- The three lines through A and B, C and D, and E and F, respectively, go through one point, denoted by '$m((A,B),(C,D),(E,F))$',
- The six points A, B, C, D, E, and F lie on one conic, denoted by '$c(A,B,C,D,E,F)$'.

We again observe the homogeneous coordinates in the plane, elements of $\mathbb{P}^3$. This means the coordinates are in $(\mathbb{R}^3\backslash\{0\})/\mathbb{R}\backslash\{0\}$, in words: all scalar multiples of a vector denote the same point. We will denote the determinant

$$\begin{vmatrix} x_A & x_B & x_C \\ y_A & y_B & y_C \\ z_A & z_B & z_C \end{vmatrix}$$

corresponding to the points A, B, and C by $[ABC]$. This notation is referred to as the *bracket notation*, a determinant $[ABC]$ as *a bracket*. In the mathematical foundation in this section parts from the masters thesis by one of Jürgen Richter-Gebert's students, Andreas Umbach, were consulted [10].

3.1 Collinearity

First, we focus on the collinearity conditions, described by $h(A,B,C)$. It is commonly known that three points A, B, and C are collinear if and only if $[ABC] = 0$.

To create a proof (as shown in the next section) we need to make a connection between several conditions. This can be done using the following theorem. In order to make reading easier, we write k for the point defined by coordinates $(x_k, y_k, z_k)^T$.

Theorem 1. *Let* $1, 2, 3, 4,$ *and* 5 *be 5 points in the plane, such that* $1, 4,$ *and* 5 *are not collinear. Then the following equivalence holds:*

$$[123] = 0 \Leftrightarrow [124][135] = [125][134]$$

Proof. The bracket is a 3-linear alternating form: Observe

$$a = [123][145] - [124][135] + [125][134].$$

Fix 1 in a. Then a is a 4-linear alternating form on $\{2, 3, 4, 5\}$. However, there is no such thing as a non degenerate 4-linear alternating form in a 3-dimensional space, so $a = 0$. Since this implies

$$[123][145] = [124][135] - [125][134]$$

the theorem is proved.

Using this theorem, we can translate a set of conditions of the form $h(A, B, C)$ to *bi-quadratic equations*:

$$[ABD][ACE] = [ABE][ACD]$$

for any D and E such that A, D, and E are not collinear.

This method of describing a geometry theorem implicitly introduces a number of non-degeneracy conditions. For example, the fact that 1, 4, and 5 are not collinear. On the one hand, this is an advantage, as we do not have to express this kind of non-degeneracy conditions explicitly. On the other hand, this is a disadvantage, as we might add some non-degeneracy conditions we are not aware of and which might be unnecessary. However, in this specific situation the disadvantage seems less important, since the user will construct a certain theorem in Cinderella, thus (in general) avoiding degenerated cases himself.

3.2 Concurrency and Conics

In this section we show how to translate the assertions $m((A, B), (C, D), (E, F))$ and $c(A, B, C, D, E, F)$ to bracket equations.

Theorem 2. *Observe the assertion* $m((A, B), (C, D), (E, F))$*. This means that the lines through A and B, C and D, and E and F go through one point. This assertion implies that all these 6 points and 3 lines are distinct, and that* *the point of concurrency is not one of A, B, C, D, E, F, thus implicitly adding non-degeneracy conditions every time we use this assertion.*

This assertion is equivalent to,

$$[ABC][CDE][EFA] = -[ABE][CDA][EFC],$$

and to

$$[ABF][CDE] = [ABE][CDF].$$

Proof. Observe the assertion $m((A, B), (C, D), (E, F))$, i.e. the lines through A and B, C and D, and E and F go through one point, say Z. This is equivalent to the combination of the three assertions

$$h(A, B, Z),\ h(C, D, Z) \text{ and } h(E, F, Z).$$

Using Theorem 1 we find the following three equations. Notice how we have to use the fact that all 6 points are distinct and none of them is the point of concurrency.

$$\begin{aligned}[ABC][AZE] &= [ABE][AZC],\\ [CDE][CZA] &= [CDA][CZE],\\ [EFA][EZC] &= [EFC][EZA]. \end{aligned} \tag{1}$$

We multiply the left- and right-hand sides and cancel terms that occur on both sides, and obtain

$$[ABC][CDE][EFA] = -[ABE][CDA][EFC], \tag{2}$$

thus proving the first equation.

As $m((A, B), (C, D), (E, F))$ is equivalent to (for example) the assertion $m((A, B), (C, D), (F, E))$ we obtain from Equation 2:

$$-[ABF][CDA][FEC] = [ABC][CDF][FEA]. \tag{3}$$

Again, we multiply the left- and right-hand sides from Equations 2 and 3 and cancel terms that occur on both sides, and we obtain

$$[ABF][CDE] = [ABE][CDF], \tag{4}$$

which proves the second equation of the theorem.

We just showed two possible encodings of the $m(..)$-assertion. However, it appears that using only the second one suffices in practice. An assertion $m(..)$ gives us only three different instances of Equation 4, all other permutations are equivalent to one of those three.

Remark 1. Because of the implicit degeneration conditions introduced, we have to be careful when using $m(..)$ as an assertion representing the thesis. Additionally, in practice it appears a proof using $m(..)$ as configuration assertions is harder to understand than a proof using $h(..)$ as configuration assertions. However, the $m(..)$-assertion still has the huge advantage that it represents three $h(..)$-assertions, thus considerably reducing the amount of configuration assertions and configuration equations.

As this shows that using the $m(..)$-assertion has both advantages we do not want to loose and disadvantages we do not want to have, we chose to leave the choice to the user of Cinderella. When trying to obtain a proof, he can decide whether he wants to use $m(..)$-assertions in the configuration, and whether he wants to use $m(..)$-assertions in the thesis. This enables the user to find the 'golden mean' between the shortness and the clarity of the proof.

The next theorem describes how to encode six points on a conic into bracket expressions.

Theorem 3. *Observe the assertion $c(A, B, C, D, E, F)$, meaning that the six points A,B,C,D, E, and F are on one conic. This assertion implies that all these six points are distinct and no three of the points are collinear, thus implicitly adding non-degeneracy conditions every time we use this assertion.*

This assertion is equivalent to the following bracket equation:

$$[ACE][BDE][ABF][CDF] = [ABE][CDE][ACF][BDF].$$

Proof. First, observe four distinct points, A, B, C, and D, and the two degenerate conics c_1 and c_2. The conic c_1 is given by the line through A and B and the line through C and D, the conic c_2 is given by the line through A and C and the line through B and D. For an arbitrary point x we have

$$\begin{array}{ll} x \in c_1 & \text{if and only if } x \text{ on } AB \text{ or } x \text{ on } CD, \text{ so } [ABx][CDx] = 0 \\ x \in c_2 & \text{if and only if } x \text{ on } AC \text{ or } x \text{ on } BD, \text{ so } [ACx][BDx] = 0. \end{array} \quad (5)$$

Now, for all $\lambda, \mu \in \mathbb{R}$, the equation

$$\lambda[ABx][CDx] + \mu[ACx][BDx]$$

describes a conic through A, B, C, and D. Now let

$$\begin{aligned} \lambda &= [ACE][BDE] \\ \mu &= -[ABE][CDE], \end{aligned} \quad (6)$$

and observe the expression

$$[ACE][BDE][ABx][CDx] - [ABE][CDE][ACx][BDx]. \quad (7)$$

Since this is a bi-quadratic expression of degree two, which evaluates to zero for $x \in \{A, B, C, D, E\}$, this defines a conic on the points A,B,C,D, and E. This means F is on that conic if and only if

$$[ACE][BDE][ABF][CDF] = [ABE][CDE][ACF][BDF], \quad (8)$$

which concludes the proof.

In general, a single $c(..)$-assertion gives us $6! = 720$ possible equations. However, in the configuration a basis of the subspace spanned by these 720 equations will suffice. Such a basis is a set of equations such that the other equations can be obtained by multiplying sides and removing pairs that occur on both sides. With basic linear algebra we can obtain a basis for this [10, p. 36]. The basis can be found in Table 1. So, every $c(..)$-assertion only adds five equations to the set of configuration equations. However, if the thesis is a $c(..)$-assertion, we have to include *all* 720 possible equations, as each of those equations is equivalent to the thesis.

$$\begin{aligned}
[ABC][ADE][BDF][CEF] &= [ABD][ACE][BCF][DEF],\\
[ABE][ACD][BDF][CEF] &= [ABD][ACE][BEF][CDF],\\
[ABE][BCD][ADF][CEF] &= [ABD][BCE][AEF][CDF],\\
[ABD][AEF][BCF][CDE] &= [ABF][ADE][BCD][CEF],\\
[ABE][ACF][BDF][CDE] &= [ABF][ACE][BDE][CDF]. \qquad (9)
\end{aligned}$$

Table 1. A basis for $c(..)$-assertions in the configuration

4 Non-Projective Geometry

In this section we present some theory that enables us to represent assertions in non-projective geometry in brackets. Someone might think that calculating with expressions that are invariant with respect to linear transformations, as described in the previous chapter, automatically makes it impossible to prove any theorems containing for example circles. This is not such a strange thought, since circles might become conics (and lose their circularity) under linear transformations. However, by adding two special points to the configuration, we can prove such theorems.

4.1 Complex Numbers

With the following procedure we can use complex numbers to express conditions involving distances, angles, etc. Given a point $P = (x, y) \in \mathbb{R}^2$ in the plane, we define $z_p \in \mathbb{C} := x + iy$. Moreover, $z_p = r \cdot e^{i\varphi}$ for certain $r, \varphi \in \mathbb{R}$. We know $z \in \mathbb{R} \Leftrightarrow z = \overline{z}$, and $z \in i\mathbb{R} \Leftrightarrow z = -\overline{z}$.

We go back to homogeneous coordinates and introduce two new 'points': $I = (i, -1, 0)$ and $J = (-i, -1, 0)$. Now observe the bracket $[ABI]$, where $A = (x_a, y_a, 1)$ and $B = (x_b, y_b, 1)$:

$$[ABI] = \begin{vmatrix} x_a & x_b & i \\ y_a & y_b & -1 \\ 1 & 1 & 0 \end{vmatrix} = x_a + iy_a - x_b - iy_b = z_a - z_b. \tag{10}$$

Likewise, the bracket $[ABJ]$ evaluates to

$$[ABJ] = \begin{vmatrix} x_a & x_b & -i \\ y_a & y_b & -1 \\ 1 & 1 & 0 \end{vmatrix} = x_a - iy_a - x_b + iy_b = \overline{z_a - z_b}. \tag{11}$$

Example 1. **(Collinearity)** Now suppose A, B, and C are collinear. Observe the complex numbers $z_1 = r_1 e^{i\varphi_1}$ of the vector $(B - A)$, and $z_2 = r_2 e^{i\varphi_2}$ of $(C - A)$. The points A, B, and C are collinear if and only if the two angles φ_1 and φ_2 are either the same or opposed to each other. This means $\varphi_1 = \varphi_2$ or $\varphi_1 = \pi + \varphi_2$, which means $z_1/z_2 \in \mathbb{R}$, or equivalently

$$\frac{B-A}{C-A} = \overline{\left(\frac{B-A}{C-A}\right)} \tag{12}$$

Using Equations 10 and 11 this is equal to

$$\frac{[BAI]}{[CAI]} = \frac{[BAJ]}{[CAJ]}, \tag{13}$$

which evaluates to

$$[ABI][ACJ] = [ACI][ABJ], \tag{14}$$

which indeed fulfills the claim at Theorem 1.

4.2 Circles

We will now show how to encode the fact that four points are on one circle in brackets.

Theorem 4. *Suppose the four points A, B, C, and D are on one circle, then the following bracket equation holds:*

$$[ACI][BDI][ADJ][BCJ] = [BCI][ADI][ACJ][BDJ].$$

This assertion will be denoted by $ci(A, B, C, D)$.

Note that this matches the bracket equation for a conic through the points A, B, C, D, I, and J (See Theorem 3).

Proof. It is a well known theorem that four points A, B, C, and D are on one circle if and only if the angle between AC and BC is equal to the angle between AD and BD. We now switch to complex numbers as described in the start of this section, and find that this is equivalent to

$$\frac{z_A - z_C}{z_B - z_C} \Big/ \frac{z_A - z_D}{z_B - z_D} \in \mathbb{R},$$

with arguing as in the example on collinearity above. This equation can be rewritten to

$$\frac{z_A - z_C}{z_B - z_C} \Big/ \frac{z_A - z_D}{z_B - z_D} = \overline{\frac{z_A - z_C}{z_B - z_C} \Big/ \frac{z_A - z_D}{z_B - z_D}}.$$

Using Equations 10 and 11 this transforms to

$$[ACI][BDI][BCJ][ADJ] = [BCI][ADI][ACJ][BDJ],$$

which concludes the proof.

5 The Prover

Suppose we are given a certain theorem in planar geometry, containing points and some collinearity conditions. This means we know that certain brackets (i.e. expressions of the form $[ABC]$) are equal to zero. We define B to be the set of all brackets, i.e. all combinations of three points from the geometry theorem, so $|B| = \binom{p}{3}$, where p denotes the number of points in the configuration.

Example 2. Suppose we have a configuration with the points A, B, C, and D. Then

$$B := \{[ABC], [ABD], [ACD], [BCD]\},$$

and $|B| = 4 = \binom{4}{3}$.

Suppose we have a geometry statement, and by Theorems 1, 2 and 3 we obtained a set of n equations following from the configuration:

$$\begin{aligned} c_{1l} &\equiv c_{1r}, \\ c_{2l} &\equiv c_{2r}, \\ &\vdots \\ c_{nl} &\equiv c_{nr}, \end{aligned} \tag{15}$$

where 'l' denotes the left hand side of the equation, and 'r' the right hand side. Each of the factors of the c_{il} and c_{ir} denotes a determinant of three points in the geometry statement, so each $c_{i,l/r}$ is a product of elements of B. Note that we use the equivalence sign '$\equiv$' rather than the normal equation sign '$=$' to make it clear that we are calculating with brackets, elements of B, rather than with the elements in $\mathbb{R}$ or $\mathbb{Q}$ they evaluate to. From Theorem 1 it follows directly that each of the factors of the c_{il} and c_{ir} is not equal to zero.

Moreover, we have an (at least one) equation that implies the thesis we want to test:

$$t_l \equiv t_r. \tag{16}$$

Note that all factors in $t_{l,r}$ should be a factor of at least one $c_{i,l/r}$. This means that the brackets in the thesis equation should occur somewhere in the configuration equations.

Remark 2. Note that it is almost always possible to express the thesis in various different equations. For the remainder of the section we will just pick one, for ease of reading. In practice we will test all of them, checking which gives us the shortest proof, if any.

Now suppose we have a certain oracle that gives us a vector $g \in \mathbb{Q}^n$, $g \neq 0$ such that

$$\frac{1}{t_l}\prod_{i=1}^{n}(c_{il})^{g_i} \equiv \frac{1}{t_r}\prod_{i=1}^{n}(c_{ir})^{g_i}. \tag{17}$$

By multiplying both sides by the greatest common divisor q of the denominators in $g_1, \ldots, g_n$, thus clearing the denominators, we obtain the following equation:

$$\left(\frac{1}{t_l}\right)^q \prod_{i=1}^{n}(c_{il})^{v_i} \equiv \left(\frac{1}{t_r}\right)^q \prod_{i=1}^{n}(c_{ir})^{v_i}, \text{ where } v_i = q \cdot g_i, \text{ so } v_i \in \mathbb{Z}. \tag{18}$$

Remark 3. In words: For each of the n equations we multiply a certain power of the left sides with each other, and the same power of the right sides. Then all terms cancel, except for $(t_l)^q$ on the left side, and $(t_r)^q$ on the right side.

By the definition of the $c_{i,l/r}$ we know

$$(c_{il})^a \equiv (c_{ir})^a \quad \forall 1 \leq i \leq n, \forall a \in \mathbb{Z}, \tag{19}$$

and since $q \neq 0$ we obtain from Equation 18:

$$\left(\frac{1}{t_l}\right)^q \equiv \left(\frac{1}{t_r}\right)^q, \quad \text{so } t_l \equiv t_r. \tag{20}$$

This means such a vector $g \in \mathbb{Q}^n$ gives us a verifiable *proof* that the thesis logically follows from the configuration. In the next section it will be shown how we can obtain such a g.

5.1 Obtaining the Proof

It will be shown that a vector g as in 17 can be found by solving linear equations. We recall that B is the set of all brackets, and define $b = |B|$ and x_i such that $\{x_1, \ldots, x_b\} = B$.

Suppose we have a configuration given by $c_{1l}, c_{1r}, \ldots, c_{nl}, c_{nr}$ and a thesis given by t_l and t_r. Recall that $c_{i,l/r}$ and $t_{l/r}$ are products of elements of B. Introduce the $b \times n$ matrix X with coefficients in $\mathbb{Z}$, defined as follows:

$$\text{for all } 1 \le k \le b, 1 \le i \le n : X_{ki} := \begin{cases} 1 & \text{if } x_k \text{ is a factor of } c_{il}, \\ -1 & \text{if } x_k \text{ is a factor of } c_{ir}, \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

The vector $Y \in \mathbb{Z}^b$ is defined in the same way from our thesis.

$$\text{for all } 1 \le k \le b : Y_k := \begin{cases} 1 & \text{if } x_k \text{ is a factor of } t_l, \\ -1 & \text{if } x_k \text{ is a factor of } t_r, \\ 0 & \text{otherwise.} \end{cases} \tag{22}$$

Now observe the following system of linear equations

$$X \cdot g = Y, \tag{23}$$

with a solution vector g. Since X and Y have integer values, we know that $g \in \mathbb{Q}^n$. It is straightforward to see that g satisfies Equation 17. Thus, we have a procedure that enables us to obtain a proof by solving linear equations, which is *much* faster than having to use Gröbner bases.

Remark 4. In general, the problem whether a geometric theorem is true or false is still equal to the decision if $t_l - t_r$ is in the ideal I generated by $c_{il} - c_{ir}$ $(i = 1, \ldots, n)$. This ideal membership problem is normally decided by means of Gröbner bases, but experimenting with Gröbner bases in this context thought us that they are too slow to be practical in our situation. However, a g as in Equation 23 shows that

$$t_l - t_r \equiv \sum_{i=1}^{n} g_i (c_{il} - c_{ir}). \tag{24}$$

which proves the ideal membership. If such a vector g does not exist however, we have no information on the ideal membership. This is why we *lose the possibility to prove a theorem to be false*, as a theorem is called 'false' only when $t_l - t_r \notin \sqrt{I}$.

6 On the Implementation

In this section some notes are given on the implementation of the prover described in Sections 3 and 4. It is meant to give an *overview* of how the transition from a geometric theorem in Cinderella to a proof of that theorem can be realized.

```
1.  A := FreePoint;
2.  B := FreePoint;
3.  a := Join(A,B);
4.  C := FreePoint;
5.  b := Join(B,C);
6.  D := FreePoint;
7.  c := Join(C,D);
8.  E := FreePoint;
9.  d := Join(D,E);
10. F := Meet(a,c);
11. e := Join(E,F);
12. G := Meet(b,d);
13. f := Join(A,G);
14. g := Join(A,D);
15. h := Join(B,E);
16. H := Meet(e,f);
17. k := Join(H,C);
```

Fig. 2. Pappos' Theorem as a Cinderella Algorithm
(Capital characters denote points, small characters denote lines)

6.1 Translating the Assertion

The translation from a geometric theorem in Cinderella into a set of assertions of the forms described in Sections 3 and 4, takes place in two steps.

Firstly, an algorithm in Cinderella (see for example Figure 2) is translated into an OpenMath `plangeo.assertion`-object. This can be done rather straightforward, as every step of the algorithm corresponds to a single OpenMath Application. For example,

```
a := Join(A,B)
```

can directly be translated into the OpenMath object below.

```
<OMA>
  <OMS name="line" cd="plangeo1"/>
  <OMV name="a"/>
  <OMA>
    <OMS name="incident" cd="plangeo1"/>
    <OMV name="a"/>
    <OMV name="A"/>
  </OMA>
  <OMA>
    <OMS name="incident" cd="plangeo1"/>
    <OMV name="a"/>
    <OMV name="B"/>
  </OMA>
</OMA>
```

Thus, walking through Cinderella's algorithm step by step, we obtain an OpenMath `plangeo1.assertion` representing the configuration. The thesis added to this object is the last non-trivial incidence the randomized prover concluded.

Cinderella is able to convert the following Cinderella algorithms into OpenMath elements: `Join`, `Meet`, `Mid`, `PointOnLine`[1], `Through`[2], `Orthogonal`, `Parallel`, `CircleMP`[3], `ConicBy5`,
`IntersectionConicLine`, `IntersectionConicConic`, `CircleBy3`, `PointOnCircle`,
`OtherIntersectionCC`[4], and `OtherIntersectionCL`[5]. Note that not all of these algorithms can be encoded to bracket equations. However, it is useful to translate as much objects as possible to OpenMath, as this OpenMath object can be used by other applications.

In this step we will restrict ourselves to elements we can translate to the assertions given in Sections 3 and 4. This means that if the OpenMath object from the previous step has elements such as `Mid`, an error will be raised and the translation will be broken off at this point. However, the OpenMath object is still valid, and might be used by an application that can handle more statements. Moreover, this two-phased design makes it possible for the prover to handle any theorem in projective geometry that can be expressed in OpenMath, not just the ones that can be constructed in Cinderella!

The `plangeo1.assertion` from the previous step is processed in the following way:

1. Find all elements (point, lines, conics) in the configuration, say $\{E_1, \ldots, E_p\}$,
2. Find all incidences, and link them to the elements, thus finding a set of incidences $F_i \subset \{E_1, \ldots, E_p\}$, where $1 \leq i \leq p$. This means that element E_i is incident to all elements in the set F_i,
3. We set $G := \{1, \ldots, p\}$,
4. While $G \neq \emptyset$:
   a) Get an index $k \in G$, where first indices corresponding to conics are processed, then circles, then points, and finally lines,
   b) Encode the element identified by k. For example: If E_k is a conic, and F_k contains 6 points, an element of the form $c(..)$ is added to the configuration,
   c) Set $G := G \backslash \{k\}$
   d) Set $F_i := F_i \backslash \{E_k\}$, for all $i \in G$,
   e) Set $G := G \backslash \{i\}$ for all $i \in G$ for which $F_i = \emptyset$.
5. Find the incidence the thesis describes. Depending on if this is an incidence between a point and a line, a point and a conic or a point and a circle, the type of the thesis will differ.

[1]A new point on an existing line
[2]A new line through an existing point
[3]MP stands for MidPoint
[4]The two intersections between two conics
[5]The two intersections between a conic and a line

Notice that the element 'points' in Item 4a may be ignored if the user stated that he does not want $m(..)$ assertions in the configuration (See Remark 1). Using the above procedure, a configuration from Cinderella is translated automatically to a configuration consisting of assertions, ready to be converted into brackets.

6.2 The Prover

The procedure explained in the previous sections was implemented in Cinderella [1], with the aid of OpenMath [5, 6, 7] and GAP [9]. A geometric theorem in Cinderella is translated into a proof in bracket algebra according to the following steps:

1. **Cinderella**: A configuration described by points and lines, some objects may have coordinates,
2. **OpenMath**: A configuration described by points and lines, as in `plangeo`, some objects may have coordinates,
3. **OpenMath**: A matrix X and a vector Y as in Equations 21 and 22, respectively,
4. **GAP**: A matrix X and a vector Y as in Equations 21 and 22, respectively,
5. **GAP**: A vector g as in Equation 17,
6. **OpenMath**: A vector g as in Equation 17,
7. **Cinderella**: A vector g as in Equation 17,
8. **Cinderella**: A string representing the proof, where the coordinates of the vector g have been translated back into their equivalents in bracket expressions.

These translations were implemented using Java, the Riaca OpenMath Library [7] and GAP [9]. The result was integrated within Cinderella and will be a part of Cinderella 2, which will be ready someday in the future with more exciting new options!

7 Examples

In this section we give some examples of geometric theorems proved using Cinderella and GAP. These theorems were created in Cinderella, 'discovered' by the internal Randomized Prover, and then, via OpenMath, given to the prover. Thus, there has been no optimization whatsoever by the user.

7.1 Pappos

We consider Pappos' Theorem, see Figure 3. The thesis is that the lines through B and C, through A and D, and through G and H go through one point (K). The full output of the prover is as follows:

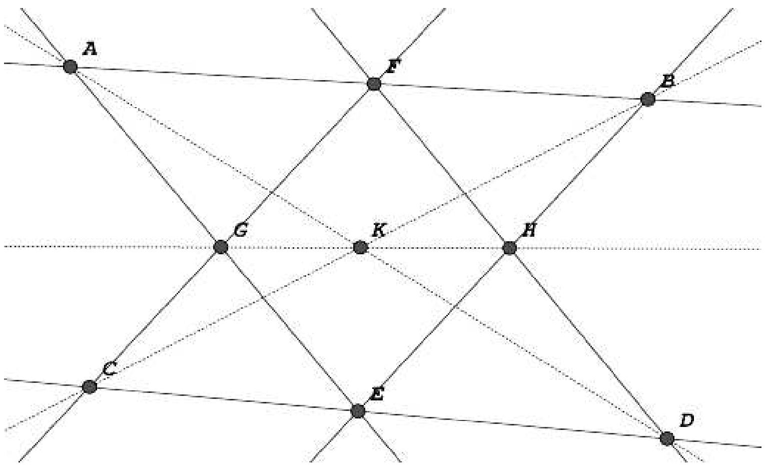


Fig. 3. Pappos' Theorem

```
Conditions:
{h(C, F, G)} On line h
{h(D, F, H)} On line g
{h(B, E, H)} On line f
{h(A, E, G)} On line e
{h(C, D, E)} On line c
{h(A, B, F)} On line a

Assertion:
{m((B, C), (A, D), (G, H))} Through point K

Number of configuration equations: 200
Number of possible theses: 3

Found a proof for thesis 1. Length: 11.
Found a proof for thesis 2. Length: 11.
Found a proof for thesis 3. Length: 10.

(1) [A.C.F][B.C.G] ==  [B.C.F][A.C.G] <== {h(C, F, G)}
(1) [B.D.F][A.D.H] ==  [A.D.F][B.D.H] <== {h(D, F, H)}
(1) [B.C.E][A.B.H] ==  [A.B.E][B.C.H] <== {h(B, E, H)}
(1) [A.B.E][B.D.H] ==  [B.D.E][A.B.H] <== {h(B, E, H)}
(1) [A.B.E][A.C.G] ==  [A.C.E][A.B.G] <== {h(A, E, G)}
(1) [A.D.E][A.B.G] ==  [A.B.E][A.D.G] <== {h(A, E, G)}
(1) [B.C.D][A.C.E] ==  [A.C.D][B.C.E] <== {h(C, D, E)}
(1) [A.C.D][B.D.E] ==  [B.C.D][A.D.E] <== {h(C, D, E)}
(1) [A.B.C][A.D.F] ==  [A.B.D][A.C.F] <== {h(A, B, F)}
(1) [A.B.D][B.C.F] ==  [A.B.C][B.D.F] <== {h(A, B, F)}
---------------------------------------------------------------------
(1) [B.C.G][A.D.H] ==  [B.C.H][A.D.G] <== {m((B, C), (A, D), (G, H))}

Checking proof... done.
Result: [B.C.G][A.D.H] == [A.D.G][B.C.H]

Found first proof in 2.08 seconds, final proof in 3.27 seconds.
```

In the remainder of the section only the proofs are given, the rest of the prover output is omitted.

7.2 Pascal

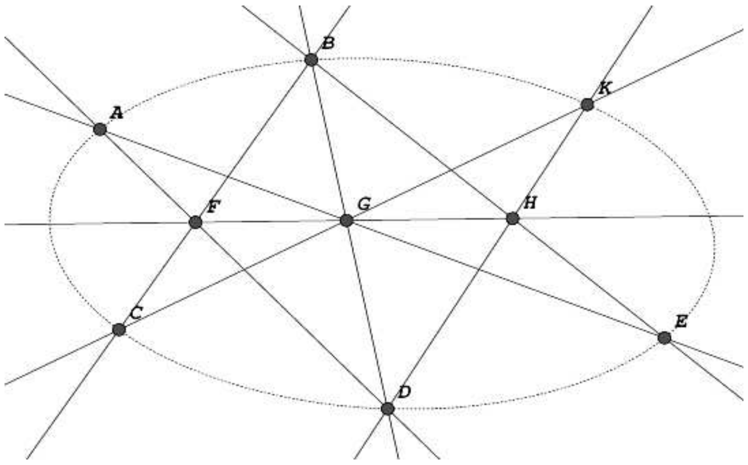


Fig. 4. Pascal's Theorem

We consider Pascal's Theorem, as shown in the picture above. The thesis is that the six points A, B, C, D, E and K are on one common conic. The proof is as follows:

```
(1)             [A.C.G][B.C.K] ==  [B.C.G][A.C.K]             <== {h(C, G, K)}
(1)             [B.D.H][A.D.K] ==  [A.D.H][B.D.K]             <== {h(D, H, K)}
(1)             [B.F.G][A.F.H] ==  [A.F.G][B.F.H]             <== {h(F, G, H)}
(1)             [B.C.G][A.B.F] ==  [A.B.C][B.F.G]             <== {h(B, C, F)}
(1)             [A.B.C][B.F.H] ==  [B.C.H][A.B.F]             <== {h(B, C, F)}
(1)             [A.B.D][A.F.G] ==  [A.D.G][A.B.F]             <== {h(A, D, F)}
(1)             [A.D.H][A.B.F] ==  [A.B.D][A.F.H]             <== {h(A, D, F)}
(1)             [A.B.E][B.C.H] ==  [B.C.E][A.B.H]             <== {h(B, E, H)}
(1)             [B.D.E][A.B.H] ==  [A.B.E][B.D.H]             <== {h(B, E, H)}
(1)             [A.C.E][A.B.G] ==  [A.B.E][A.C.G]             <== {h(A, E, G)}
(1)             [A.B.E][A.D.G] ==  [A.D.E][A.B.G]             <== {h(A, E, G)}
-----------------------------------------------------------------------------------------
(1) [A.C.E][A.D.K][B.C.K][B.D.E] ==  [B.D.K][B.C.E][A.D.E][A.C.K] <== {co(A, B, C, D, E, K)}
```

7.3 Miguel

Miguel states that if $ABCF$, $BCDE$, $CEFG$, $AFGH$ and $ABDH$ form five circles, then the four points D, E, G and H are on one circle, see Figure 5. This is proved as follows:

```
(1) [A.F.I][F.G.H][A.H.J][G.I.J] ==  [A.F.H][F.G.I][A.I.J][G.H.J] <== {ci(A, F, G, H)}
(1) [A.F.H][A.I.J][F.G.J][G.H.I] ==  [A.F.J][A.H.I][F.G.H][G.I.J] <== {ci(A, F, G, H)}
(1) [C.E.I][C.F.J][E.G.J][F.G.I] ==  [C.E.J][C.F.I][E.G.I][F.G.J] <== {ci(C, E, F, G)}
(1) [B.C.I][B.D.J][C.E.J][D.E.I] ==  [B.C.J][B.D.I][C.E.I][D.E.J] <== {ci(B, C, D, E)}
(1) [A.B.H][B.D.I][A.I.J][D.H.J] ==  [A.B.I][B.D.H][A.H.J][D.I.J] <== {ci(A, B, D, H)}
(1) [A.B.J][A.H.I][B.D.H][D.I.J] ==  [A.B.H][A.I.J][B.D.J][D.H.I] <== {ci(A, B, D, H)}
(1) [A.B.I][B.C.F][A.F.J][C.I.J] ==  [A.B.F][B.C.I][A.I.J][C.F.J] <== {ci(A, B, C, F)}
```

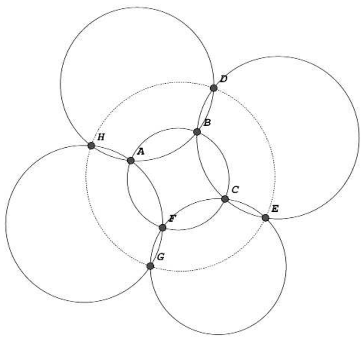


Fig. 5. Miguel's six circle theorem

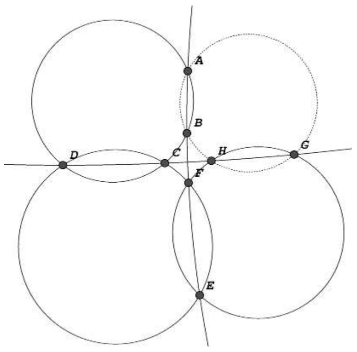


Fig. 6. Another six circle theorem

```
(1) [A.B.F][A.I.J][B.C.J][C.F.I] ==  [A.B.J][A.F.I][B.C.F][C.I.J] <== {ci(A, B, C, F)}
--------------------------------------------------------------------------------------
(1) [D.E.I][D.H.J][E.G.J][G.H.I] ==  [G.H.J][E.G.I][D.H.I][D.E.J] <== {ci(D, E, G, H)}
```

7.4 Six circles

Observe the geometric configuration in Figure 6. The thesis is that the points A, B, G and H are on one circle. Although this theorem is a lot like Miguel's theorem about six circles, thisone can not be proved to be true by our prover. In [10] Umbach gives a proof in 4 steps, using human reasoning about this configuration. He too, however, is unable to prove this theorem automatically.

8 Conclusion

We made it possible to obtain proofs for theorems put together in Cinderella by any user. However, we must be careful not to forget that we do not have the possibility to prove theorems false, as we did when using Gröbner bases. Sure, the prover can handle *some* theorems in projective and non-projective geometry, but often fails, as for example in Section 7.4. However, in exchange for these disadvantages, we gained the possibility to proof geometric theorems considerably faster than before. Moreover, these proofs are short and, unlike proofs made by Gröbner bases, easy to check by hand.

In the course of the project the power of OpenMath became clear. Because of the existing link between OpenMath and GAP [7] it was extremely easy to use GAP for solving linear equations without any additional programming. Although that is not such a difficult algorithm, there is no need to implement it yourself. The added advantage is that GAP will perform a lot better than our own home-made algorithm. Other advantages of the extensive use of

OpenMath include the possibility to reuse intermediate results, import geometric theorems made by hand, and in the future, use other computer algebra packages than GAP, or import geometric theorems made by other geometry programs. All this can be done rather easily, because of the clarity of the OpenMath standard.

A few questions remain open on proving geometry theorems using bracket algebra. For example, we again consider the fact that, when proving a geometric theorem, we are actually testing ideal membership (see Remark 4). In practice however, we are able to find a proof for a fairly large number of geometric theorems, using only linear combinations. The question why we 'get lucky' so often remains open. The power of the prover may be extended in the future, for example using methods proposed recently in the Journal of Symbolic Computation by Li and Wu [3, 4].

Cinderella and its randomized prover have been out there for a few years already, the invariant theory used exists since the twenties, and solving linear equations is not the most recent discovery either. However, the combination of these three items into a single program gives us a rather interesting result. OpenMath made it possible to do this in a structured and extendable manner, helping to achieve the goal of this project. It is now possible to create geometric theorems by pointing and clicking, and then automatically obtain a proof for that theorem. Not only can this proof be obtained very quickly, it is short and easy to check!

References

1. The Interactive Geometry Software Cinderella. `http://www.cinderella.de`.
2. Ulrich Kortenkamp. *Foundations of Dynamic Geometry.* PhD thesis, Swiss Federal Institute of Technology, Zurich, 1999. `http://www.cinderella.de/papers/diss.pdf`.
3. Hongbo Li and Yihong Wu. Automated short proof generation for projective geometric theorems with Cayley and bracket algebras: I. Incidence geometry. *Journal of Symbolic Computation*, 36(5):717–762, 2003.
4. Hongbo Li and Yihong Wu. Automated short proof generation for projective geometric theorems with Cayley and bracket algebras: II. Conic geometry. *Journal of Symbolic Computation*, 36(5):763–809, 2003.
5. OpenMath. `http://www.openmath.org`.
6. OpenMath Content Dictionary: plangeo1..5. This CD defines symbols for planar Euclidean geometry. `http://www.win.tue.nl/∼amc/oz/om/cds/geometry.html`.
7. The Riaca OpenMath Library. `http://www.riaca.win.tue.nl`.
8. Jürgen Richter-Gebert. Mechanical theorem proving in projective geometry. *Annals of Mathematics and Artificial Intelligence*, 13:139–172, 1995.

9. Martin Schönert et al. *GAP – Groups, Algorithms, and Programming.* Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, fifth edition, 1995.
10. Andreas Umbach. Automatisches erzeugen geometrischer beweise. Master's thesis, Institut für Theoretische Informatik ETH Zürich, 2000.

Towards Solving the Dynamic Geometry Bottleneck Via a Symbolic Approach (extended abstract)

F. Botana and T. Recio*

Departamento de Matemática Aplicada I, Universidad de Vigo, Campus A Xunqueira, 36005 Pontevedra, Spain
`http://rosalia.uvigo.es/fbotana`
Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria, Santander, Spain
`http://www.recio.tk`

1 Introduction

In the late eighties, two computer programs allowing dynamic changes in plane geometric constructions were simultaneously introduced: Cabri [14] and The Geometer's Sketchpad, GSP [11]. The key feature of this software is that unconstrained parts of the construction can be dragged on by the user and, as they move, all other elements in the construction automatically self–adjust, preserving mutual dependency relations and constraints [12]. Because of its evident impact in computer aided mathematics instruction, a specific name, ie. dynamic geometry, was soon coined for programs presenting this kind of feature. Dynamic geometry software offers a virtual environment where accurate construction of geometric configurations can be carried out. Besides Cabri and GSP, Cinderella [17, 13] and Geometry Expert, GEX [7] can be also quoted as well known and performing dynamic geometry environments.

On the other hand, even in this summary introduction to the topic, we would like to highlight three problems (creating, perhaps, a real bottleneck for future developments) that dynamic geometry software has to face in general: the so called continuity problem, the loci generation and the proof capability. The problem of continuity appears when small changes in some element of the construction involve sudden changes in some other parts of the construction, as it happens in the construction shown in Fig. 1: The common point P of circles C_1 and C_2 suddenly jumps when dragging the circle C_1 along the line AB. Although the mathematical theory behind Cinderella has solved this

*Second author supported by grant BFM2002-04402-C02-02 from the Spanish MCyT and by the University of Vigo, on a sabbatical visit

problem, it must be noted that the symbolic approach taken in GDI gives another way of circunventing the continuity issue.

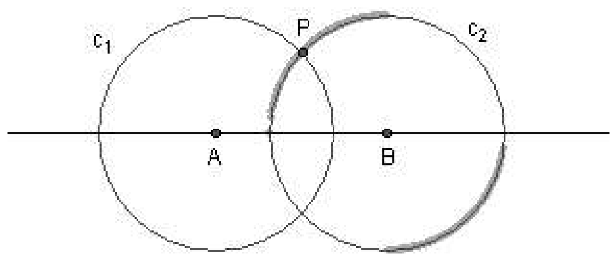


Fig. 1. A case of discontinuity

Regarding loci generation, dynamic geometry environments heavily rely on an interactive approach [1]. Thus, in order to build up some geometric locus, they allow the user to drag an element of a construction, usually a point, and then to visualize the path of some other element, whose trace is supposed to be activated by the displacement of the first object. This approach, quite successful in many instances, impedes the computation of loci defined through *a posteriori* conditions and, even worse, the loci equations cannot be determined in general for further computation purposes.

Finally, a common use of dynamic geometry environments such as Cabri and GSP, in elementary geometry teaching and learning, involves an activity that has been termed as ‘visual proving’ (of properties and theorems). The numerical accuracy of the constructions and the possibility to experiment with different instances (by dragging the basic geometric objects in a given statement) get the user convinced about the truth or falsity of some conjecture. Partly reacting to this approach, last generation programs such as Cinderella, GEX and GEOTHER [18] start including formal tools for automatic geometric theorem proving.

In this context, the goal of this short note is to give some simple examples from a prototype of a recent program, GDI [2, 3], a Spanish acronym of Intelligent Dynamic Geometry. Apart from being a standard dynamic geometry environment, GDI includes enhanced tools for loci generation and automatic proving, plus another distinguished feature, namely, a discovery option allowing a user to finding complementary hypotheses for arbitrary statements to become true, or, in other words, to finding the missing hypotheses so that a given conclusion follows from a given (perhaps drastically) incomplete set of hypotheses.

The key technique for all these improvements is the development of an automatic “bridge” between the graphic and the algebraic counterparts of the program, as proposed in [16, 15].

2 Loci discovery

In order to describe a geometry problem by a finite set of polynomials, we only consider problems involving incidence, congruence and parallelism relations. Given a geometric construction with a point whose locus is the one we are looking for, the procedure begins by translating the geometric properties into algebraic expressions, after selecting a coordinate system. We use the field of rational numbers Q and C, the field of complex numbers, as an algebraically closed field containing the former. The collection of construction properties is then expressed as a set of polynomial equations

$$p_1(x_1,\dots,x_n)=0,\dots,p_r(x_1,\dots,x_n)=0,$$

where $p_1,\dots,p_r \in Q[x_1,\dots,x_n]$. Thus, the affine variety defined by $V = \{p_1=0,\dots,p_r=0\} \subset C^n$ contains all points $(x_1,\dots,x_n) \in C^n$ which satisfy the construction requirements, that is, the set of all common zeros of $p_1,\dots,p_r$ in the n-dimensional affine space of C describe all the possible positions of the construction points. In particular, the positions of the locus point define the locus we are searching for. Thus, supposing that the locus point coordinates are x_{n-1}, x_n, the projection

$$\pi_{n-2} : V \subset C^n \to C^2$$

gives an extensional definition of the locus in the affine space C^2. This projection can be computed via the $(n-2)$th elimination ideal of $\langle p_1,\dots,p_r\rangle$, I_{n-2}. The Closure theorem states that $V(I_{n-2})$ is the smallest affine variety containing $\pi_{n-2}(V)$, or, more technically, that $V(I_{n-2})$ is the Zariski closure of $\pi_{n-2}(V)$. So, except some missing points that lie in a variety strictly smaller than $V(I_{n-2})$, we can describe the locus computing a basis of I_{n-2}. This basis is computed as follows: given the ideal $\langle p_1,\dots,p_r\rangle \subset Q[x_1,\dots,x_n]$, let G be a Groebner basis of it with respect to lex order where $x_1 > x_2 > \cdots > x_n$. The Elimination theorem states that $G_{n-2} = G \cap Q[x_{n-1},x_n]$ is a Groebner basis of I_{n-2}.

As an illustration of the above procedure, GDI will be used to discover a recent generalization of Wallace–Steiner theorem [9, 8]: Given a triangle ABC and three directions, not all three equal, nor parallel to the triangle sides, find the locus of points X such that its projections M, N, P along the three directions determine a triangle of oriented area k (Fig. 2). Once the geometric construction is done, the user imposes the condition that the oriented area of MNP is, say, 1, area$(M,P,N) = 1$. The geometric predicates are automatically translated by GDI into polynomials, and CoCoA [4] is used (in the background) to perform the elimination task. Finally, the locus equation is returned to the dynamic environment, where the curve is plotted (Fig. 3)

Note that since the locus equation is now known by the system, this new object can be used to construct new objects. For instance, if the imposed condition was the alignment of M, N, P, GDI could compute the envelope of lines MNP, a tricuspidal hypocycloid.

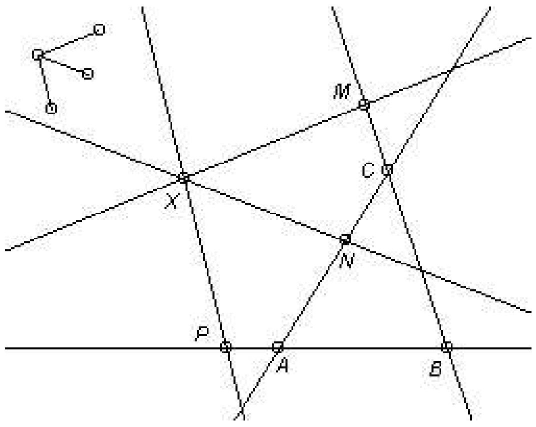


Fig. 2. Find the locus of X such that the oriented area of MNP is constant

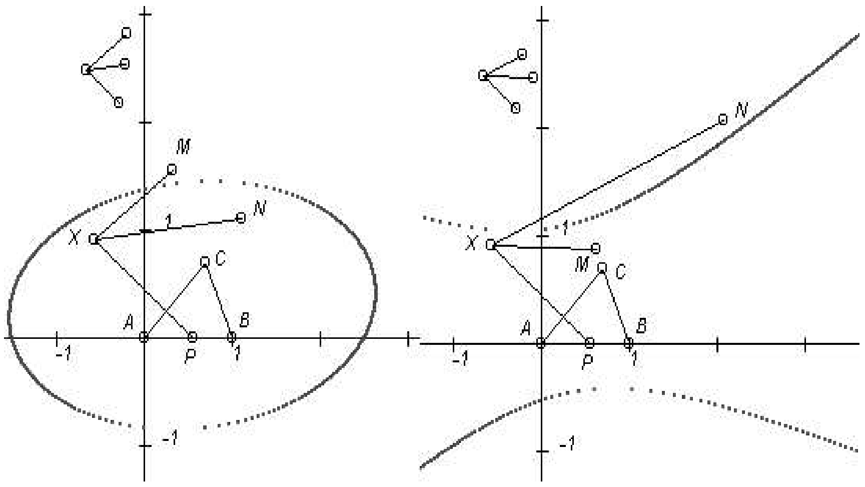


Fig. 3. An ellipse ($43890x2 - 16139xy + 143719y2 - 43890x - 76139y - 165360 = 0$) and a hyperbola ($64470x2 + 87313xy - 521163y2 - 64470x + 323036y + 532680 = 0$) in Giering–de Guzmán's theorem

3 The Algebraic Approach to Automatic Theorem Proving

For the last 20 years, symbolic algebraic techniques have been successfully used for automatically proving theorems in elementary geometry (see [19] for an exhaustive repository of related papers). The practical interest of this goal (i.e to automate, through the algebraic translation of hypotheses and theses, theorem proving) could be related, for instance, to its potential applications in geometric constraint solving and parameterized CAD [10, 5, 6].

The algebraic approach roughly proceeds as follows. A geometric statement (a finite set of hypotheses and a thesis) is translated into two multivariate polynomial systems, H, T. The statement is declared to be true if the hypotheses variety is contained in the thesis variety, $Var(H) \subseteq Var(T)$. Different approaches exist to test this inclusion in commutative/algebraic geometry, mainly Wu–Ritt characteristic sets and Gröbner bases. Moreover, it usually happens that the inclusion does not happen because of some small set of points; in this case the algebraic approach takes care of detecting the degenerate cases that should be removed from the hypothesis. Both methods work in an algebraically closed field, so the decision about the truth of a geometric statement involves not only real solutions, but the complex ones also.

In GDI, following this approach, and through the cooperation of the graphic environment with a symbolic computation program (such as Mathematica or CoCoA, at user's choice), the user can state and prove (opening a suitable Menu) or disprove different conjectures on a given construction. Moreover it can discover complementary hypothesis (but not only for degenerate cases, when we have a nearly true statement, but in generally false statement), following the approach of [16]. The following example shows how this is done through some Internet application (but, of course, it can be also locally computed).

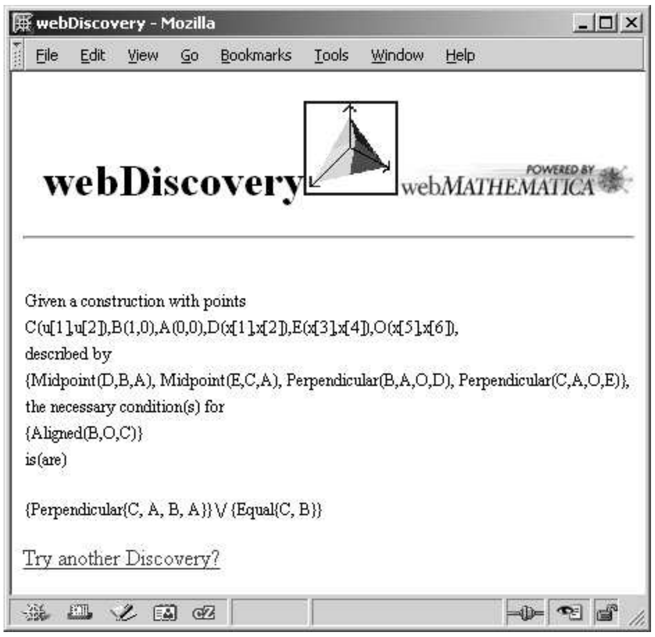
4 webDiscovery

The GDI symbolic algorithms have been used to develop an Internet application. webDiscovery (`rosalia.uvigo.es/sdge/web/2D`) is an open web–based tool for automatic discovery in elementary Euclidean geometry. It accepts user–defined geometric constructions, which are uploaded to a Java Servlet server, where two computer algebra systems, CoCoA and Mathematica, return the discovered facts about the construction. As a simple illustration, we consider a triangle ABC and its circumcircle O. In order to discover the necessary conditions for the collinearity of B, O and C, the user uploads a text file, that is output by GDI as a by-product of the construction, as follows

```
Points
C(u[1],u[2])
B(1,0)
A(0,0)
D(x[1],x[2])
E(x[3],x[4])
O(x[5],x[6])
LingProperties
Midpoint(D,B,A)
Midpoint(E,C,A)
Perpendicular(B,A,O,D)
Perpendicular(C,A,O,E)
```

```
LingConditions
Aligned(B,O,C)
DiscProperties
```

and the application finds as conditions the rightness of the angle BAC and a degenerated condition (Fig. 4).



Given a construction with points
C(u[1],u[2]),B(1,0),A(0,0),D(x[1],x[2]),E(x[3],x[4]),O(x[5],x[6]),
described by
{Midpoint(D,B,A), Midpoint(E,C,A), Perpendicular(B,A,O,D), Perpendicular(C,A,O,E)},
the necessary condition(s) for
{Aligned(B,O,C)}
is(are)

{Perpendicular{C, A, B, A}} \/ {Equal{C, B}}

Fig. 4. The necessary conditions for the alignment of the circumcenter

5 Conclusions

Perhaps the most astonishing fact is that GDI performs quite well for an endless collection of examples, from trivial facts to rather complicated explorations of new results –too long to be detailed here–, due to the high performance of the current algebraic elimination engines and to the careful connection of the graphic and symbolic engines. In the near future, GDI and webDiscovery will be able to export/accept geometric constructions coded in OpenMath.

References

1. Botana, F.: Interactive versus symbolic approaches to plane loci generation in dynamic geometry environments. *Proc. I Int. Workshop on Computer Graphics and Geometric Modelling* (CGGM'2002), LNCS, 2330, 211–218 (2002)
2. Botana, F., Valcarce, J.L.: A software tool for the investigation of plane loci. *Mathematics and Computers in Simulation*, 61(2), 141–154 (2003)
3. Botana, F., Valcarce, J.L.: Automatic determination of envelopes and other derived curves within a graphic environment. *Mathematics and Computers in Simulation*, to appear
4. Capani, A., Niesi, G., Robbiano, L.: CoCoA, a system for doing Computations in Commutative Algebra. Available via anonymous ftp from: `cocoa.dima.unige.it`
5. Gao, X.S., Chou, S.C.: Solving geometric constraint systems. I. A global propagation approach. *Computer–Aided Design*, 30, 47–54 (1998)
6. Gao, X.S., Chou, S.C.: Solving geometric constraint systems. II. A symbolic approach and decision of Rc–constructibility. *Computer–Aided Design*, 30, 115-122 (1998)
7. Gao, X.S., Zang, J.Z., Chou, S.C: Geometry expert (Nine Chapters Publ., Taiwan, 1998)
8. Giering, O.: Affine and projective generalization of Wallace lines. *Journal of Geometry and Graphics*, 1(2), 119–133 (1997)
9. de Guzmán, M.: An extension of the Wallace–Simson theorem: projecting in arbitrary directions. *American Mathematical Monthly*, 106(6), 574–580 (1999)
10. Hoffmann, C., Bouma, W., Fudos, I., Cai, J., Paige, R.: A geometric constraint solver. *Computer–Aided Design*, 27, 487–501 (1995)
11. N. Jackiw, The Geometer's Sketchpad (Key Curriculum Press, Berkeley, 1997)
12. King, J., Schattschneider, D.: Geometry turned on (Mathematical Association of America, Washington DC, 1997)
13. Kortenkamp, U.: Foundations of dynamic geometry, Ph.D. Thesis, ETH, Zurich, 1999.
14. Laborde, J.M., Bellemain, F.: Cabri Geometry II (Texas Instruments, Dallas, 1998)
15. Recio, T.: Cálculo simbólico y geométrico (Síntesis, Madrid, 1998)
16. Recio, T., Vélez, M. P.: Automatic discovery of theorems in elementary geometry. *Journal of Automated Reasoning*, 23, 63–82 (1999)
17. Richter–Gebert, J., Kortenkamp, U.: The interactive geometry software Cinderella (Springer, Berlin, 1999)
18. Wang, D.: GEOTHER: A geometry theorem prover. *Proc. 13th International Conference on Automated Deduction* (CADE 1996), LNCS, 1104, 166–170 (1996)
19. `http://www-calfor.lip6.fr/~wang/GRBib/Welcome.html`

How to construct an elliptic curve with a given number of points (extended abstract)

Reinier Bröker

reinier@math.leidenuniv.nl
Mathematisch Instituut, Universiteit Leiden,
Postbus 9512, 2300 RA Leiden
The Netherlands

Introduction

The popularity of elliptic curves in the algorithmic community has greatly increased during the past few decades due to their applications to factoring, primality testing and cryptography. The Hasse theorem from 1934 already states that for an elliptic curve E defined over the finite field $\mathbb{F}_q$, the order of its point group $E(\mathbb{F}_q)$ is an element of the 'Hasse interval'

$$\mathcal{H}_q = [q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$$

around q. There are several algorithms that compute the order of $E(\mathbb{F}_q)$ efficiently, i.e. polynomial in $\log q$. A mathematical natural question to ask is the 'inverse problem' of point counting, a problem for which no efficient algorithm is currently known:

Problem. *Given an integer $N \geq 1$, find a finite field $\mathbb{F}_q$ and an elliptic curve $E/\mathbb{F}_q$ for which the number of $\mathbb{F}_q$-rational points equals N.*

Clearly, a necessary condition for this problem to be solvable is that N is contained in some Hasse interval $\mathcal{H}_q$. We would therefore like that prime powers q are not that far apart, i.e. we want that $\bigcup_q \mathcal{H}_q = \mathbb{Z}_{\geq 1}$. The contribution of the true prime powers to this union is negligible, so we may as well restrict to a prime field $\mathbb{F}_q$. Although we then know that all integers in the Hasse interval do occur as the group order of $E(\mathbb{F}_q)$ for some curve $E/\mathbb{F}_q$, we can't prove that for given N we can actually find a prime q such that $N \in \mathcal{H}_q$. The reason is that it can't be proven at the moment that the gap between two consecutive primes q_1 and q_2 is less than $4\sqrt{q_1}$. From a practical point of view however there are lots of primes q for which $N \in \mathcal{H}_q$, since we expect (by the prime number theorem) that in the neighbourhood of N one out of every $\log N$ integers is prime.

A first naïve approach to solving our problem might be to simply pick a prime q such that $N \in \mathcal{H}_q$ and write down curves over $\mathbb{F}_q$ until we find one having N points. Although the time spent per curve is very small, we expect that we need to try about $\sqrt{N}$ curves before finding a correct one. In practice, it turns out that this algorithm becomes quite impractical if $N \gg 10^{15}$.

CM-approach

We can do considerably better than this 10^{15} with the deterministic algorithm described in this section. If we take any q for which $N \in \mathcal{H}_q$ and write $N = q+1-t$, then the Frobenius morphism F_q

$$F_q : E \to E \qquad (x,y) \mapsto (x^q, y^q)$$

of our desired curve E satisfies the quadratic relation

$$F_q^2 - tF_q + q = 0$$

of discriminant $\Delta = t^2 - 4q < 0$. For given q, specifying N is therefore the same as specifying the trace t of Frobenius, i.e. specifying the subring $\mathbb{Z}[F_q] \subseteq \mathrm{End}(E)$. We want to find an elliptic curve E whose endomorphism ring *contains* the Frobenius morphism of trace t. The first step of the algorithm will be to select the prime q that we will use, which will also determine $\Delta = \Delta(q) = (q-1-N)^2 - 4N$. If we would take a prime q such that N is at the end of $\mathcal{H}_q$ we expect $\Delta \approx N^{1/2}$. But if we also remove the square factors of Δ by varying q a bit, we can often achieve much better than $N^{1/2}$.

For the rest of this abstract, we assume that q is fixed. If we write $N = 1+q-t$ again, then the miraculous fact is that for $t \neq 0$ (which ensures that $\mathrm{End}(E)$ is an order in the imaginary quadratic field $\mathbb{Q}(\sqrt{D})$), we can lift E *together* with its Frobenius to characteristic 0. More precisely, there is a number field $H \supseteq \mathbb{Q}(\sqrt{\Delta})$ and an elliptic curve $\tilde{E}/H$ such that

- • there is $\varphi_q \in \mathrm{End}(\tilde{E})$ satisfying $\varphi_q^2 - t\varphi_q + q = 0$;
- • q splits completely in $H/\mathbb{Q}$ and for every prime $\mathfrak{q}|q$ in H the reduced curve $\tilde{E}$ mod $\mathfrak{q}$ is an elliptic curve having $q+1-t$ points.

It turns out that we can actually compute the j-invariant of this curve $\tilde{E}$. Namely, if we denote the discriminant of $\mathbb{Q}(\sqrt{\Delta})$ by D and compute a set $\mathrm{Cl}(D)$ of reduced binary quadratic forms of discriminant D, then $j(\tilde{E})$ is a root of the following polynomial:

$$f_D = \prod_{[a,b,c] \in \mathrm{Cl}(D)} \left(X - j\left(\frac{-b+\sqrt{D}}{2a} \right) \right) \in \mathbb{Z}[X].$$

The j-function used in the expression above is now the well known modular function $\mathbb{H} \to \mathbb{C}$ with q-expansion $j(q) = \frac{1}{q} + 744 + 196884q + \ldots$ and f_D is in fact the minimal polynomial for $j(\tilde{E})$. Since f is actually a polynomial with *integer* coefficients, we can approximate $j((-b+\sqrt{D})/2a)$ upto high enough precision, expand the product above and then round the coefficients to the nearest integer.

Knowing the minimal polynomial for $j(\tilde{E})$, we can reduce it modulo q to obtain $\overline{f}_D \in \mathbb{F}_q[X]$ which splits completely (an excellent check of our computations so far). Its roots are the j-invariants of the curves over $\mathbb{F}_q$ having N points. Given a root j, we put $k = 27j/(4(j-1728))$ and the curve defined by $y^2 = x^3 - kx - k$ has j-invariant j. Either that curve or its quadratic twist has N points.

Whereas the naïve algorithm would take hours to construct an elliptic curve having 10^{15} points, this complex analytic algorithm only takes a few seconds.

A non-archimedean approach

One problem with the previous algorithm however is that we have to compute $j((-b+\sqrt{D})/2a)$ with very high accuracy because rounding errors occur when we expand the polynomial f_D. Since rounding errors are inherent in working over the complex numbers, we propose the following solution: embed the number field H of the previous section in a p-adic field $\mathbb{Q}_p$ instead of in $\mathbb{C}$! We must then find a substitute for the computation of the '$j((-b+\sqrt{D})/2a)$' of the previous section, since the computation via the q-expansion has no meaning over $\mathbb{Q}_p$.

The first step is to choose the p that we will use for the rest of the algorithm. We can embed H into $\mathbb{Q}_p$ if and only if p splits completely into $H/\mathbb{Q}$ i.e. if p splits into two principal ideals in $\mathbb{Q}(\sqrt{D})$. This in turn is equivalent to the existence of an elliptic curve $E_p/\mathbb{F}_p$ with Frobenius of trace u satisfying $u^2 - 4p = Dk^2$ for some integer k. By construction this includes our desired curve E, so we could take $p = q$ here. But of course we want to keep p as small as possible, so given D we find the smallest u such that $u^2 - D = 4p$. Once we have found our p, we use the naïve algorithm from the introduction to actually construct an elliptic curve E_p with Frobenius of trace u. (Usually $p \ll N^{1/2}$ so this is a cheap operation.)

Just as in the previous section, we want to *lift* this curve E_p (with its Frobenius) to a curve $\tilde{E}_p$ which will be defined over $\mathbb{Q}_p$. This lift is actually unique and is called the *canonical lift* of E_p. We define a lift $E_p^1/\mathbb{Q}_p$ of $E_p/\mathbb{F}_p$ by lifting the coefficients of the Weierstrass equation to elements of $\mathbb{Q}_p$. A trivial, but crucial, observation is that E_p^1 is the canonical lift accurate upto 1 p-adic digit. There is an entire set S of curves over $\mathbb{Q}_p$ which have this property. In [1], we give the following algorithm to compute the canonical lift. We view S as a subset of $\mathbb{C}_p$ via the j-function and we give an *analytic* map $\rho : S \to S$ which has the canonical lift as its *fixed point*. Moreover, we can compute the derivative at the canonical lift $\tilde{E}_p$, which enables us to use the Newton iteration process

$$j(E_p^{k+1}) = j(E_p^k) - \frac{j(\rho(E_p^k)) - j(E_p^k)}{\rho(\tilde{E}_p) - 1} \qquad k \in \mathbb{Z}_{\geq 1}$$

to converge to the j-invariant of $\tilde{E}_p$ itself. The computation of $\rho(E_p^k)$ is a non-trivial matter which involves the extensive use of modular and division polynomials. After some optimalisation steps however, the algorithm in [1] is quite fast in practice.

Knowing the j-invariant of the canonical lift, we still have to compute the other roots of f_D. These are the j-invariants of the curves which are isogenous to $\tilde{E}_p$. We compute them by employing the modular polynomials once more. Once we have all the roots, we expand the polynomial f_D and proceed as in the complex analytic approach.

Further developments

There is still one problem with the complex analytic and the p-adic CM-algorithm: the coefficients of the polynomial f_D are huge. Not only do they grow exponentially with D, but also 50 digit numbers appear already for moderately small discriminants. As was already noted by Weber in the early 1900's, we can decrease the size of the coefficients considerably by employing other functions than the j-function. For instance, the j-function has a holomorphic cube root γ_2 and if $D \neq 0 \bmod 3$ we can compute a polynomial $f_D^{\gamma_2}$ very similar to f_D. The polynomial $\overline{f_D^{\gamma_2}} \in \mathbb{F}_q[X]$ also splits completely and its roots are cube roots of the j-invariants of the curves having N points.

For this γ_2 the decrease in the size of the coefficients of f_D is a factor three, but we can do even better than that. If the discriminant D is also 1 modulo 8, we can gain a factor 72 by using the classical Weber $\mathfrak{f}$. For other discriminants there are other functions for which the gain is about a factor of 20.

The theory of computing the polynomials f_D^g for functions $g \neq j$ is firmly rooted in complex analytic arguments. Much of it can however be made to work in a non-archimedean setting. A first approach is given in [1] where it is explained how to use γ_2 and $\mathfrak{f}$ in a p-adic setting. This time we put an extra structure on the 3- resp. the 48-torsion, which enables us to work with these functions in a p-adic setting. Just as for the j-function we have *modular polynomials*, which are currently the main algorithmic tool for computing f_D^g. To illustrate the power of our method, we constructed (in about 20 minutes on a normal PC) an elliptic curve having exactly 10^{30} points: if we take $q = 10^{30} + 1999999999167681$, then the elliptic curve defined by

$$y^2 = x^3 + 669397215131271955483581235905x + 363369366443977510319399421188$$

over $\mathbb{F}_q$ has exactly 10^{30} rational points.

Reference

1 Bröker, R. and Stevenhagen, P.: *Elliptic curves with a given number of points* to appear in Algorithmic Number Theory Symposium VI, copyright Springer-Verlag, 2004,
preprint: `http://www.math.leidenuniv.nl/reports/2003-21.shtml`

Computing the Stratification of Actions of Compact Lie Groups

Thomas Bayer

bayert@in.tum.de
Institut für Informatik
Technische Universität München
Boltzmannstr. 3, 85748 Garching, Germany
http://www14.in.tum.de/personen/bayert

Summary. We provide a constructive approach to the stratification of the representation- and the orbit space of linear actions of compact Lie groups contained in $\mathrm{GL}_n(\mathbb{R})$ on $\mathbb{R}^n$ and we show that any d-dimensional stratum, respectively, its closure can be described by d sharp, respectively, relaxed polynomial inequalities and that d is also a lower bound for both cases. Strata of the representation space are described as differences of closed sets given by polynomial equations while d-dimensional strata of the orbit space are represented by means of polynomial equations and inequalities. All algorithms have been implemented in SINGULAR V2.0.

Introduction

In 1983 Abud and Sartori [1] pointed out the relation between spontaneous symmetry breaking and stratifications of linear actions of compact Lie groups and presented several applications in particle physics. Spontaneous symmetry breaking can briefly be described as follows. Let G be a compact Lie group which acts linearly on $\mathbb{R}^n$, let $\varphi_0 \in \mathbb{R}^n$ be the ground state of a physical system and let $V_\gamma(z)$ be a G-invariant potential which determines φ_0 and depends on the parameter γ. Varying γ might change φ_0 into φ_0' and the stabilizer group $G_{\varphi_0'}$ of φ_0' may be "smaller" than the stabilizer G_{φ_0} (i.e., moving from φ_0 to φ_0' amounts to a loss of symmetry), which can be seen as a breaking of symmetry. In this way various patterns of spontaneous symmetry breaking occur, which correspond to distinct phases of the model. It is well-known (see for instance [15]) that the orbit space $\mathbb{R}^n/G$ is a semialgebraic set and there exists a disjoint decomposition of $\mathbb{R}^n/G$ in finitely many semialgebraic sets, called strata, whereas any stratum consists of points of the same symmetry type. The knowledge of a description of each stratum in terms of polynomial equations and inequalities is important for numerous applications (e.g., construction of invariant potentials, symmetric bifurcation theory, see [1], [4], [9], [10]).

There are several approaches for constructing the stratification of the orbit space of a compact Lie group[1] starting with Abud and Sartori, see [1], while Gatermann [9] provides a systematic exposition for compact Lie groups.

These algorithms (except [4], [5]) construct a stratification of the orbit space $\mathbb{R}^n/G$ of a compact Lie group G by using the matrix $\mathrm{grad}(z)$ which is defined on $\mathbb{R}^n/G$. We propose a different approach, namely, to compute a stratification of the representation space of G, and only then to construct the stratification of the orbit space (or the images of relevant strata) by means of elimination theory (equations) and refinements of results of Procesi and Schwarz (inequalities), see [15]. Additional, our algorithms describe any d-dimensional stratum and its closure by at most d inequalities, which turns out to be optimal. This approach has several advantages compared to the present approach[2], namely: Primary decomposition is done before the (non-linear) Hilbert map is applied, no superfluous components in the orbit space are computed, the association of strata and their stabilizers is quite obvious and, finally, it is possible to compute only those strata, which are relevant for the application under consideration. We also show how to compute inequalities which describe a stratum only up to generic equivalence but contain fewer terms. For several applications, like the construction of continuous potentials on the orbit space, this approach may lead to easier computations. For polynomial potentials, inequalities need not be calculated since the Zariski-closure of a stratum suffices.

In addition, we show that each d-dimensional stratum, respectively its closure, can be presented by at most d strict, respectively relaxed, inequalities and that d is also a lower bound.

1 On Invariant Theory of Compact Lie Groups and Orbit Spaces

We present some background on invariants of compact Lie groups and orbit spaces. In both sections we use fundamental facts from semialgebraic geometry like the Tarksi-Seidenberg principle, for which we refer to [7]. For short, an *basic open* (*basic closed*) *semialgebraic subset* of the algebraic set $V \subseteq \mathbb{R}^n$ is of the form $\{v \in V \mid g_i(v) > 0, 1 \leq i \leq r\}$, respectively, $\geq$ instead of $>$, where $g_1, g_2, \dots, g_r \in \mathbb{R}[x_1, x_2, \dots, x_n]$. In the sequel we call an inequality of the form $f > 0$, respectively, $f \geq 0$ strict, respectively, relaxed. An *open*

[1]Explicit algorithms for finite groups, which yield a minimal number of inequalities, are given in [5].

[2]Equations for the (Zariski-closure) of strata are computed out of rank conditions on the matrix $\mathrm{grad}(z)$. The locus where $\operatorname{rank} \mathrm{grad}(z) \leq d$ contains all d-dimensional strata of the orbit space and must be decomposed in irreducible components in order to obtain equations defining these strata. Some of these components may be superfluous, i.e., $\mathrm{grad}(z)$ is not positive semidefinite for some points.

(*closed*) *semialgebraic subset* of V is a finite union of basic open (basic closed) semialgebraic subsets of V.

1.1 Invariants of Lie Groups

Let G be a compact Lie group and $\rho : G \to \mathrm{GL}_n(\mathbb{R})$ be a faithful representation. In the sequel we identify G and its image $\rho(G) \subset \mathrm{GL}_n(\mathbb{R})$. It is well-known that $\mathbb{R}^n$ admits a G-invariant scalar product $(\_,\_)_G$ on $\mathbb{R}^n$ (see for instance [8]). By the Gram-Schmidt orthonormalization process there exists $A \in \mathrm{GL}_n(n)$ such that $A \cdot G \cdot A^{-1} \subseteq O_{\mathbb{R}}$, i.e, the representation ρ is equivalent to an orthogonal representation. From now on we assume $G \subseteq O_{\mathbb{R}}$ and that G acts as usual on $\mathbb{R}^n$. In the sequel let $\mathbb{K}$ be one of the fields $\mathbb{R}$ or $\mathbb{C}$. For $X \subseteq \mathbb{K}^n$ we define $\mathcal{I}(X) := \{f \in \mathbb{K}[t_1, t_2, \ldots, t_n] \mid f(x) = 0 \text{ for all } x \in X\}$, the *ideal* of X and for an ideal $I \subseteq \mathbb{K}[x_1, x_2, \ldots, x_n]$ we define $\mathcal{V}(I) := \{x \in \mathbb{K}^n \mid f(x) = 0 \text{ for } f \in I\}$, the *variety* associated to I. A subset $U \subseteq \mathbb{K}^n$ is *closed* in the Zariski topology if and only if $U = \mathcal{V}(I)$ for some ideal $I \subseteq \mathbb{K}[x_1, x_2, \ldots, x_n]$. A polynomial $f \in \mathbb{K}[x_1, x_2, \ldots, x_n]$ is *invariant* w.r.t. G if $f(g^{-1} \cdot \mathbf{x}) = f(\mathbf{x})$ for all $g \in G$. The ring $\mathbb{K}[x_1, x_2, \ldots, x_n]^G$, consisting of all invariant polynomials w.r.t. G, is called the *invariant ring* of G (ρ will be omitted). By Hilbert's Finiteness Theorem, the invariant ring is finitely generated as a $\mathbb{K}-$algebra. Homogeneous generators $\pi_1, \pi_2, \ldots, \pi_m$ of $\mathbb{K}[x_1, x_2, \ldots, x_n]^G$ are called *fundamental invariants* (i.e., each invariant polynomial is a polynomial in $\pi_1, \pi_2, \ldots, \pi_m$). Fundamental invariants define the projection

$$\begin{aligned} \pi : \mathbb{K}^n &\longrightarrow \mathbb{K}^n/G \subseteq \mathbb{K}^m \\ \mathbf{x} &\longmapsto (\pi_1(\mathbf{x}), \pi_2(\mathbf{x}), \ldots, \pi_m(\mathbf{x})) \end{aligned}$$

of $\mathbb{K}^n$ onto an embedding of the orbit space $\mathbb{K}^n/G \subseteq \mathbb{K}^m$, also called the *Hilbert map*. Note that π maps closed sets to closed sets[3] and that each fiber contains precisely one closed orbit (see for instance[13]). For $\mathbb{K} = \mathbb{C}$ the image of $\pi(\mathbb{C}^n) \subseteq \mathbb{C}^m$ equals the variety of the ideal of relations of $\pi_1, \pi_2, \ldots, \pi_m$ (see for instance[13]). Over $\mathbb{R}$ it is well-known that the image of π is a semialgebraic set.

Proposition 1. *Let $G \subset \mathrm{GL}_n(\mathbb{R})$ be a compact Lie group. The orbit space $\mathbb{R}^n/G$ of G is a semialgebraic set semialgebraically homeomorphic to $\pi(\mathbb{R}^n)$.*

Proof. It is well-known that the orbits of G can be separated by fundamental invariants of G (see for instance Theorem 3.4.3. in [14]). By the Tarski-Seidenberg principle (see for instance [7]) the real image of π is a semialgebraic set (it equals the projection of the graph, which is a real algebraic set). □

[3]Note that the map π is proper.

Note that the orbit space of an algebraic group parameterizes all closed orbits. Hence the orbit space of a compact Lie group G parameterizes all orbits of G since they are closed. Orbits which are not closed cannot be separated by polynomials so group actions having non-closed orbits cannot be stratified by using their invariant rings, see [16].

1.2 Inequalities defining Orbit Spaces

Procesi and Schwarz have constructed polynomial inequalities which have to be added to the equations coming from the Hilbert map of a compact Lie group G, which need not be a subgroup of $O_n(\mathbb{R})$, in order to describe an embedding of the quotient $\mathbb{R}^n/G \subset \mathbb{R}^m$. Essential parts of the proof are the existence of a closed orbit in each fiber of π (see for instance [13]) and the existence of a G-invariant inner product $(\_,\_)$ on $\mathbb{R}^n$, which is used to construct the $m \times m$ matrix $\operatorname{grad}(v) = (d\pi_i(v), d\pi_j(v))_{i,j=1,\dots,m}$ for $v \in \mathbb{C}^n$ where $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ is the Hilbert map. Here we have used the identification[4] of $\mathbb{R}^n$ with its dual $\operatorname{Hom}(\mathbb{R}^n, \mathbb{R})$. They proved that a point $z \in \mathcal{V}(I)$, where $I \subset \mathbb{R}[z_1, z_2, \dots, z_m]$ is the ideal of relations among $\pi_1, \pi_2, \dots, \pi_m$, lies in $\mathbb{R}^n/G$ if and only if the matrix $\operatorname{grad}(z)$ is positive semidefinite. The constraint that $\operatorname{grad}(z)$ must be semidefinite yields inequalities for describing $\mathbb{R}^n/G$. Recall that the type of a real $m \times m$ Matrix M equals (p, q) where p, respectively, q denote the number of positive, respectively, negative eigenvalues counted with multiplicities. Obviously, $\operatorname{rank} M = p + q$.

Proposition 2. *An $m \times m$ matrix M over $\mathbb{R}$ is positive semidefinite (denoted by $M \geq 0$) iff all symmetric minors of M are non-negative. The matrix M is positive definite (denoted by $M > 0$) iff all principal minors of M are positive.*

Proof. We refer to, e.g., Section IX.72 in [18]. □

In order to define the matrix $\operatorname{grad}(z)$ on the orbit space we have to show that all entries are invariant w.r.t. G. By $d\pi(z)$ we denote the Jacobian matrix of π at z.

Proposition 3. *Let $G \subset \operatorname{GL}_n(\mathbb{R})$ be a compact Lie group. For $\sigma \in G_v$ the Jacobian of the Hilbert map $\pi : \mathbb{R}^n \to \mathbb{R}^n/G$ satisfies $d\pi(v) = d\pi(v) \circ \sigma$. In particular, the functions $v \mapsto \operatorname{grad}(v)_{ij}$ are invariant.*

Proof. Follows from $\pi(v) = \pi(\sigma \cdot v)$, the chain rule, and the fact that σ is linear. □

Therefore the matrix $\operatorname{grad}(v)$ is also defined on $\mathbb{K}^n/G \subseteq \mathbb{K}^m$ and can be extended to the whole of $\mathbb{K}^m$. Procesi and Schwarz provided the following description of the orbit space.

[4]Note that $d\pi_j$ is a differential form, so $d\pi_j(z) : \mathbb{R}^n \to \mathbb{R}$ is a linear form.

Theorem 1. *(Procesi-Schwarz [15]) Let $G \subset \mathrm{GL}_n(\mathbb{R})$ be a compact Lie group and let $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$ be such that $\pi_1, \pi_2, \ldots, \pi_m$ generate $\mathbb{R}[x_1, x_2, \ldots, x_n]^G$. The quotient space is given by*

$$\mathbb{R}^m/G = \pi(\mathbb{R}^n) = \{z \in \mathbb{R}^m \mid \mathrm{grad}(z) \geq 0, z \in \mathcal{V}(I)\}$$

where $I \subset \mathbb{R}[y_1, y_2, \ldots, y_m]$ is the ideal of relations of $\pi_1, \pi_2, \ldots, \pi_m$.

Proof. We refer to [15]. □

Inequalities for the orbit space can be obtained from the condition $\mathrm{grad}(z) \geq 0$. This can be checked by means of Proposition 1.2.2, i.e., testing if all $2^n - 1$ symmetric minors of $\mathrm{grad}(z)$ are ≥ 0. In subsequent sections we use the theorem of Procesi and Schwarz and a modification of Decartes rule of signs to provide an optimal description[5] of the orbit space and all of its strata and their closures (defined in the following section), which are useful for several applications.

Example 1. Consider the action of the compact Lie group $G = O_2 \subset GL_2(\mathbb{R})$ on $\mathbb{R}^4$, given by $(g \cdot x, g \cdot y), g \in G, x, y \in \mathbb{R}^2$, and its complexification $G_{\mathbb{C}}$ (see Section 3.1). We may choose three algebraically independent fundamental invariants $\pi_1 = t_1^2 + t_2^2, \pi_2 = t_1 t_3 + t_2 t_4, \pi_3 = t_3^2 + t_4^2$. The invariant ring of G, respectively, $G_{\mathbb{C}}$ equals $\mathbb{K}[t_1, t_2, t_3, t_4]^G = \mathbb{K}[\pi_1, \pi_2, \pi_3]$ where $\mathbb{K} = \mathbb{R}$, respectively, $\mathbb{K} = \mathbb{C}$. The Hilbert map is $\pi = (\pi_1, \pi_2, \pi_3) : \mathbb{K}^4 \to \mathbb{K}^3$. Since π_1, π_2, π_3 are algebraically independent, we obtain $\mathbb{C}^4/G_{\mathbb{C}} = \mathbb{C}^3 = \mathrm{im}(\pi)$. Over the reals, we apply Theorem 1.2.1 and Proposition 2.2.10 to the matrix $\mathrm{grad}(z) = \begin{pmatrix} 4z_1 & 2z_2 & 0 \\ 2z_2 & z_1 + z_3 & 2z_2 \\ 0 & 2z_2 & 4z_3 \end{pmatrix}$ and obtain the description

$$\mathbb{R}^4/G = \mathrm{im}(\pi) = \left\{ \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} \in \mathbb{R}^3 \;\middle|\; \begin{array}{l} z_1 + z_3 \geq 0, z_1^2 - 2z_2^2 + 6z_1 z_3 + z_3^2 \geq 0, \\ z_1^2 z_3 + z_1 z_3^2 - z^2(z_1 + z_3) \geq 0 \end{array} \right\} \subsetneq \mathbb{R}^3.$$

Remark 1. (a) For practical purposes the dependence on a G-invariant scalar product may be problematic.
(b) It is not necessary that $G \subseteq O_{\mathbb{R}}$ for computing inequalities if a G-invariant inner product is given in an effective form.

2 On the Stratification of the Representation and Orbit Space

Consider a compact Lie group $G \subset \mathrm{GL}_n(\mathbb{R})$, the set of points having the same symmetry type w.r.t. G form a partition of $\mathbb{R}^n$ in finitely many distinct

[5]The description is optimal in the number of inequalities , i.e., we show that this number is an upper and lower bound.

open sets, also called a stratification. We present underlying definitions and properties of of strata and their closures (Zariski- or Euclidean topology). These properties will be used in subsequent sections to compute equations and inequalities for describing strata and their closures.

2.1 On the Stratification of the representation- and orbit space

We provide the definition of strata, respectively, stratifications and associated objects like orbit type, etc. In the sequel $G \subset \mathrm{GL}_n(\mathbb{R})$ denotes a compact Lie group and $\mathrm{cl}_Z(X)$, respectively, $\mathrm{cl}_E(X)$ denote the closure of the set X in the Zariski, respectively, Euclidean topology.

Definition 1. *Let $E \subseteq \mathbb{R}^n$ be a semialgebraic set. A stratification of E is a finite partition E_λ of E where each E_λ is a semialgebraically connected locally closed*[6] *equidimensional semialgebraic subset (or a finite set of points) of $\mathbb{R}^n$ such that $E_\lambda \cap \mathrm{cl}_E(E_\beta) \neq \emptyset$ and $\lambda \neq \beta$ implies $E_\lambda \subset E_\beta$ and $\dim E_\lambda < \dim E_\beta$. For $\lambda \in \Lambda$ the set E_λ is called a stratum and $\mathrm{cl}_E(E)_\lambda$ is called a semi-stratum of the stratification, and if $d = dim E_\lambda$ then E_λ is called a $d-$stratum.*

Given $x \in \mathbb{R}^n$, the set $G(x) = \{g \cdot x \mid g \in G\}$ is called the *orbit* of x and the group $G_x = \{g \in G \mid g \cdot x = x\}$ is called the *stabilizer* of x.

Proposition 4. *Let G be an algebraic group (defined over the field $\mathbb{K}$) which acts algebraically (via α) on $\mathbb{K}^n$. For $x \in \mathbb{K}^n$ the stabilizer G_x and the set $X_d = \{x \in X \mid \dim G_x \geq d\}$ are closed.*

Proof. Let $\pi_2 : X \times X \to X$ be the projection onto the second component, $i_x : G \hookrightarrow G \times X, i_x(g) = (g, x)$ be an injection for $x \in X$ and define $\alpha' : G \times X \to X \times X$ by $\alpha'(g, x) = (\alpha(g, x), x)$. All maps are continuous (w.r.t. the Zariski-topology), hence the fibers of $\pi_2 \circ \alpha' \circ i$ are closed. The stabilizer of x is closed since G_x is isomorphic to $\alpha'^{-1}(x, x) = \{(g, x) \mid \alpha(g, x) = x\}$. We also obtain that $X_d = \{x \in X \mid \dim(\pi_2 \circ \alpha' \circ i)^{-1}(x) \geq d\}$ hence the claim follows from upper-continuity of the fiber dimension. □

Definition 2. *For a subgroup $H \subseteq G$ we denote the conjugacy class of H in G by by $[H] = \{gHg^{-1} \mid g \in G\}$. The orbit type of $x \in \mathbb{R}^n$ is $[x] := [G_x]$. For $u, v \in \mathbb{R}^n$ we define $[u] < [v]$ if $G_u \subset H$ for some $H \in [v]$. The associated stratum, respectively, semi-stratum of $[x]$ is $\Sigma_x := \{y \in \mathbb{R}^n \mid [x] = [y]\}$, respectively, $\mathrm{cl}_E(\Sigma_x)$.*

The orbit type is a measure for the symmetry of the points of $\mathbb{R}^n$. We have $[x] > [y]$ if the point x has more symmetries than the point y, i.e., $gG_yg^{-1} \subset G_x$ form some $g \in G$. The notation of strata is justified by the fact that these sets, respectively, their images under the Hilbert map form a stratification of the representation-, respectively, orbit space.

[6]The set E_λ is open in its metric closure $\mathrm{cl}_E(E_\lambda)$.

Proposition 5. *Let $G \subset \mathrm{GL}_n(\mathbb{R})$ be a compact Lie group.*

(a) There are only finitely many different orbit types, i.e,. $\{[G_x] \mid x \in \mathbb{R}^n\}$ is finite.
(b) The orbit types form a lattice.
For $v \in \Sigma_p := \{x_0 \in \mathbb{R}^n \mid \operatorname{rank} d\pi(x)x_0$ is maximal$\}$ the orbit type $[v]$ is the least element.
(c) For each $v \in \mathbb{R}^n$ there exists a small neighborhood $U \subset \mathbb{R}^n$ of v such that $u \in U$ implies $[u] \leq [v]$.

Proof. (a) see for instance Ch. IV.10 in [8].
(b) Note that $\operatorname{rank} d\pi(x)v$ is maximal iff $\dim N_v^0$ is maximal (see Section 2.2) hence the stabilizer of v is contained in $[w]$ for all $w \in \mathbb{R}^n$.
(c) We refer, e.g., to [1].

□

The set Σ_p, which is dense in $\mathbb{R}^n$, is called the *principal stratum* of G.

Proposition 6. *Let $G \subset \mathrm{GL}_n(\mathbb{R})$ be a compact Lie group.*

(a) For a subgroup $H \subseteq G$ of G the set $\mathbb{R}^n_H = \{x \in \mathbb{R}^n \mid H \subseteq G_x\}$ is a vectorspace. In particular, the set $\{x \in \mathbb{R}^n \mid G_x = H\}$ is Zariski-open in $\mathbb{R}^n_H$.
(b) For $0 \neq x \in \mathbb{R}^n$ each stratum Σ_x is open in its closure (both metric and Zariski) and $G(x)$ is a proper subset of Σ_x.

Proof. (a) Let $x, y \in \mathbb{R}^n_H$ and $g \in H$. Obviously, $g \cdot (x + y)$ and $g \cdot \lambda x, \lambda \in \mathbb{R}$, are contained in $\mathbb{R}^n_H$. The set $S = \{x \in \mathbb{R}^n_H \mid G_x \supset H\}$ is of dimension less than $\mathbb{R}^n_H$ and can be written as the union of all strata Σ_y with $[y] > [H]$ intersected with $\mathbb{R}^n_H$. By Proposition 2.1.5, the set S is closed, hence $\mathbb{R}^n_H \setminus S$ is Zariski-open.
(b) The first claim follows from Theorem 2.2.2. For the second claim note that $G(x)$ is compact, hence the set $\{\lambda x \mid \lambda \in \mathbb{R}, \lambda > 0\}$ is not contained in $G(x)$ but in Σ_x.

□

Note that the closure of a stratum of the representation space need not be a finite union of vectorspaces, as it is the case for finite groups, see Example 3.4.4. We conclude this section by giving a description of the orbit space (and its stratification) in terms of equations and relaxed inequalities obtained from Procesi's and Schwarz's Theorem. Here strata are described as differences of closed semialgebraic sets.

Corollary 1. *Let $G \subset \mathrm{GL}_n(\mathbb{R})$ be a compact Lie group, let $x \in \mathbb{R}^n$ and $y = \pi(x)$.*

(a) Let $\Sigma_x \subseteq \mathbb{R}^n$ be a stratum. Then $\mathrm{cl}_E(\hat{\Sigma}_y) = \pi(\mathrm{cl}_E(\Sigma_x)) = \{z \in \mathbb{R}^m \mid \mathrm{grad}(z) \geq 0, z \in \mathcal{V}(J)\}$ where $J \subset \mathbb{R}[z_1, z_2, \ldots, z_m]$ is the ideal of the image of Σ_x under π.

(b) *Let* $\mathrm{cl}_E(\Sigma_x) = \Sigma_x \cup B_x$ *be a disjoint union (*B_x *is a finite union of lower-dimensional strata). Then* $\hat{\Sigma}_x = \pi(\Sigma_x) = \pi(\mathrm{cl}_E(\Sigma_x)) - \pi(B_x)$*, i.e.,*

$$\hat{\Sigma}_x = \{z \in \mathbb{R}^m \mid z \in \mathrm{cl}_Z(\pi(\Sigma_x)), z \notin \pi(B_x), \mathrm{grad}(z) \geq 0\}$$

2.2 Properties of Strata

We describe properties of strata and semi-strata on the representation and orbit space. In the representation space closures of strata, respectively, strata can be described by closed sets, respectively, differences of closed sets. For a description of the orbit space Procesi and Schwarz have derived the condition that $\mathrm{grad}(z) \geq 0$ (see Theorem 1.2.1), but they only provide the criterium given in Proposition 1.2.2, which yields $2^d - 1$ inequalities (provided that d equals the dimension of the orbit space). These inequalities may also be used to describe all topological closures of strata on the orbit space and therefore also all strata by forming differences of closed sets (see Corollary 2.1.1). We show that a d-dimensional stratum respectively, its closure can be described by d sharp, respectively, relaxed inequalities and the ideal of its Zariski-closure in $\mathbb{R}^n/G$ and that d is also a lower bound. In particular, we provide effective descriptions relying on equations and inequalities.

The stratification of the representation space of a compact Lie group is completely determined by the matrix $d\pi(x)v$. Since $\mathbb{R}^n$ admits a G-invariant inner product $(\_,\_)_G$ we may define the orthogonal complement N_v to $T_v(G(v))$ and the decomposition $N = N_v^0 \oplus N_v^1$, where $N_v^0 = \{w \in N_v \mid w \text{ is } G_v\text{-invariant}\}$ and N_v^1 is the orthogonal complement of N_v^0 in N_v. Note that G need not be a subgroup of the orthogonal group.

Proposition 7. *Let* $G \subset \mathrm{GL}_n(\mathbb{R})$ *be a compact Lie group. We have*

$$\ker d\pi(x)x_0 = T_{x_0}G(x_0) \oplus N_{x_0}^1 \text{ and } \mathrm{im} d\pi(x)x_0 \cong N_{x_0}^0.$$

Proof. Note that $v \in T_{x_0}G(x_0)$ implies $v \in \ker d\pi(x_0)$ since π is G-invariant. Let V be the the vectorspace generated by the gradients (considered as elements of $\mathbb{R}^n$) $d\pi_1(x_0), d\pi_2(x_0), \ldots, d\pi_m(x_0)$, i.e., $V = \mathrm{im}\ d\pi(x_0)$. Note that $v \in \ker d\pi(x_0)$ implies $d\pi_i(x_0) \cdot v = 0$ so $v \in N_{x_0}$. By Proposition 2.2.3 we have $d\pi_i(x_0) \circ \sigma = d\pi_i(x_0)$ for $\sigma \in G_v$, hence $V \subseteq N_v^0$. Now $v \in N_{x_0}^0 \setminus V$ implies $v \in \ker d\pi(x_0)$. Hence the rank of the matrix $d\pi(x_0)$ augmented by the column v equals the rank of $d\pi(x_0)$ and so $v \in V$. □

Proposition 8. *Let* $G \subset \mathrm{GL}_n(\mathbb{R})$ *be a compact Lie group. We have*

$$T_{x_0}\Sigma_{x_0} = T_{x_0}G(x_0) \oplus N_{x_0}^0.$$

In particular, $T_{\pi(x_0)}\hat{\Sigma}_{x_0} \cong N_{x_0}^0$.

Proof. One has to show that any curve through x_0 and contained in Σ_{x_0} has a tangent vector at x_0 which is contained in $T_{x_0}G(x_0) \oplus N_{x_0}^0$. This proof can be found in Section V of [1]. □

Corollary 2. *We have* $\dim \Sigma_{x_0} = \dim T_{x_0} + \dim N^0_{x_0} = \dim G - \dim G_{x_0} + \dim N^0_{x_0}$ *and* $\dim \hat{\Sigma}_{\pi(x_0)} = \dim N^0_{x_0}$.

Theorem 2. *Let* $G \subset \mathrm{GL}_n(\mathbb{R})$ *be a compact Lie group and* $\pi : \mathbb{R}^n \to \mathbb{R}^n/G \subseteq \mathbb{R}^m$ *be the Hilbert map.*

(a) The union $\Sigma^{(d)}$ *of all strata whose image under* π *is of dimension* d *equals the open semi-algebraic set*

$$\Sigma^{(d)} = \{v \in \mathbb{R}^n \mid \operatorname{rank} d\pi(v) = d\}\,.$$

(b) The union Σ^d *of all strata whose image under* π *is of dimension at most* d *equals the closed semi-algebraic set*

$$\Sigma^d = \{v \in \mathbb{R}^n \mid \operatorname{rank} d\pi(v) \leq d\}$$

In addition, $\mathrm{cl}_Z(\Sigma^{(d)}) = \mathrm{cl}_E(\Sigma^{(d)}) = \Sigma^d$.

Proof. (a) Note that a stratum is a smooth semi-algebraic set, so by Proposition 2.2.8 we have $\operatorname{rank} d\pi(v) = \dim \operatorname{im}(d\pi(v)) = \dim T_{\pi(v)}\hat{\Sigma}_{\pi(v)} = \dim \hat{\Sigma}_{\pi(v)}$.
(b) The set Σ^d can be defined by the vanishing of all $(d{+}i)\times(d{+}i)$ minors of $\frac{\partial \pi}{\partial x}$ where $i \geq 1$. If $d \geq \min\{n, m\}$ then $\Sigma^d = \mathbb{R}^n$. Note that $\Sigma^{(d)} = \Sigma^d \setminus \Sigma^{d-1}$. □

So far we have only considered semistrata, respectively, strata on the representation space. Unfortunately, we need at most $2^n - 1$ inequalities, obtained from the symmetric minors of grad(z). A direct description of a d-dimensional stratum by means of equations and (strict) inequalities can be obtained from the constraint that the type of grad(z) equals $(d, 0)$. We apply Decartes rule of sign to the characteristic polynomial of the matrix grad(z) in order to obtain an optimal number of inequalities.

A sequence $a_0, a_1, \ldots, a_n$ has a sign change if there exists i, j s.t. $a_i a_{i+j} < 0$ and $a_i a_{i+k} \geq 0$ for $1 \leq k < j$. For a polynomial $f = \sum_{i=0}^n a_i t^i$ we define the number of sign changes $N_+(f)$ respectively alternative sign changes $N_-(f)$ by the total number of sign changes of the sequence $a_0, a_1, \ldots, a_n$ respectively of the sequence $a_0, -a_1, a_2, \ldots, (-1)^i a_i, \ldots, (-1)^n a_n$. By $Z_+(f)$ respectively $Z_-(f)$ we denote the number of positive respectively negative real roots of f.

Proposition 9. *(Descartes rule of sign; see [18]) Let* $f \in \mathbb{R}[t]$ *be a nonzero polynomial. There exist* $\rho_+, \rho_- \in \mathbb{N}$ *s.t.* $N_+(f) = Z_+(f) - 2\rho_+$ *and* $N_-(f) = Z_-(f) - 2\rho_-$. *Moreover, if* f *has only real roots then* $N_+(f) = Z_+(f)$ *and* $N_-(f) = Z_-(f)$.

We state a refinement of a well-known result in matrix analysis (see for instance Ch. 7 in [12]).

Corollary 3. *Let $M \in \mathrm{Mat}_n(\mathbb{R})$ be a symmetric matrix of* $\operatorname{rank} M = d > 0$ *and $p(t) = \sum_{i=0}^{n} a_i t^i$ be its characteristic polynomial. Then M is of type $(d, 0)$ iff $(-1)^i a_{n-i} > 0$ for $1 \leq i \leq d$.*

Proof. Note that $a_{n-d-1} = \ldots = a_0 = 0$ and all roots of $p(t)$ are real. By Proposition 2.2.9 we have $N_+(p) = Z_+(f)$ as required. □

By relaxing all inequalities obtained from conditions about sign changes of the characteristic polynomial we obtain a criterium for positive semidefiniteness without assumptions about the rank. This yields an upper bound for the description of closures of strata.

Proposition 10. *Let $M \in \mathrm{Mat}_n(\mathbb{R})$ be a symmetric matrix and $p(t) = \sum_{i=0}^{n} a_i t^i$ be its characteristic polynomial. Then M is positive semidefinite iff $(-1)^i a_{n-i} \geq 0$ for $1 \leq i \leq n$.*

Proof. Let M be a symmetric matrix of $\operatorname{rank} M = d > 0$ having a negative eigenvalue. Note that $a_{n-d-1} = a_{n-d-2} = \ldots = a_0 = 0$ and $a_n = 1$. By Decartes rule of sign (Proposition 2.2.9) there exists a minimal $i > 0$ s.t. $(-1)^n(-1)^{n-i} a_i < 0$. For n even we obtain $(-1)^{n-i} a_i < 0$ a contradiction to $(-1)^{n-i} a_{n-i} \geq 0$ since $(-1)^i = (-1)^{n-i}$. In case n odd the sign change gives $(-1)(-1)^{n-i} a_{n-i} = (-1)^{n-i+1} a_{n-i} < 0$, a contradiction to $(-1)^i a_{n-i} = (-1)^{n-i+1} a_{n-i} \geq 0$. □

Theorem 3. *Let $G \subset \mathrm{GL}_n(\mathbb{R})$ be a compact Lie group, let $\pi_1, \pi_2, \ldots, \pi_m$ be fundamental invariants of G and let I be their ideal of relations. Let $d \leq \dim \mathbb{R}^n/G$ be an integer and I_d be the ideal of all $d \times d$ minors of* $\operatorname{grad}(z)$*. By $p_d(t) = \sum_{i=0}^{m} (-1)^{m-i} \delta_i t^i$ we denote the characteristic polynomial of* $\operatorname{grad} z$ *modulo I_d.*

(a) We have $\{z \in \mathcal{V}_{\mathbb{R}}(I) \mid \operatorname{grad}(z) \geq 0, \operatorname{rank} \operatorname{grad}(z) = d\} = \{z \in \mathcal{V}_{\mathbb{R}}(I_d) \mid \delta_1(z) > 0, \delta_2(z) > 0, \ldots, \delta_d(z) > 0\}$..

(b) Relaxing the strict inequalities in part (a) gives the set $\{z \in \mathcal{V}_{\mathbb{R}}(I) \mid \operatorname{grad}(z) \geq 0\}$.

(c) Let J be the ideal of the Zariski-closure of a d-dimensional stratum $\hat{\Sigma}_d$ and $\delta'_i = \delta_i \bmod J$. Then $\hat{\Sigma}_d = \{z \in \mathcal{V}_{\mathbb{R}}(J) \mid \delta'_1(z) > 0, \delta'_2(z) > 0, \ldots, \delta'_d(z) > 0\}$. For the topological closure of $\hat{\Sigma}_d$ we obtain $\mathrm{cl}_E(\hat{\Sigma}_d) = \{z \in \mathcal{V}_{\mathbb{R}}(J) \mid \delta'_1(z) \geq 0, \delta'_2(z) \geq 0, \ldots, \delta'_d(z) \geq 0\}$.

(d) Let J be the ideal of the Zariski-closure of a d-dimensional stratum $\hat{\Sigma}_d$ and suppose that $\operatorname{grad}(z)$ *is so arranged that the first d principal minors do not vanish identically on $\hat{\Sigma}_d$. Then $\hat{\Sigma}_d$ is generically equivalent (the symmetric difference has codimension at least* 1*) to $\{z \in \mathcal{V}_{\mathbb{R}}(J) \mid \Delta_1(z) > 0, \Delta_2(z) > 0, \ldots, \Delta_d(z) > 0\}$ where $\Delta_1, \Delta_2, \ldots, \Delta_d$ are the first d principal minors of* $\operatorname{grad}(z)$*.*

(e) Suppose that $\pi_1, \pi_2, \ldots, \pi_d$ are algebraically independent. The principal stratum of $\mathbb{R}^n/G$ is given by $\hat{\Sigma}_p = \{z \in \mathbb{R}^d \mid \Delta_1(z) > 0, \Delta_2(z) > 0, \ldots, \Delta_d(z) > 0\}$ where $\Delta_1, \Delta_2, \ldots, \Delta_d$ are all principal minors of $\operatorname{grad}(z)$*.*

Proof. Part (a),(b), and (c) follow from Proposition 2.2.9. For Part (d) note that $\Delta_i(z) = 0$ defines a hypersurface in $\hat{\Sigma}_d$. Part (e) follows from $I = \{0\}$, i.e, $\mathcal{V}(I) = \mathbb{R}^d$ and from $\operatorname{rank}\operatorname{grad}(z) = d$ for all $z \in \hat{\Sigma}_p$. □

For a given d-dimensional stratum respectively its topological closure, the number of d inequalities obtained from the previous theorem is optimal, as shown by the following example.

Example 2. Let $G \subset \mathrm{GL}_n(\mathbb{R})$ be the finite group generated by all $n \times n$ diagonal matrices of the form $(1, 1, \ldots, 1, -1, 1, \ldots, 1)$. Fundamental invariants are given by $t_1^2, t_2^2, \ldots, t_n^2$. Hence the orbit space is the positive orthant $z_1 \geq 0, z_2 \geq 0, \ldots z_n \geq 0$ and any d-dimensional stratum respectively its topological closure is given by equations $z_{i_1} = \ldots = z_{i_{n-d}} = 0$ and inequalities $z_{i_{n-d+1}} > 0, \ldots, z_{i_n} > 0$ respectively $\leq$ instead of $>$, where $i_1, i_2, \ldots, i_n$ is a permutation of $1, 2, \ldots, n$. It is well-known that any such set cannot be described by fewer than d inequalities (see for instance [7]).

We obtain the following geometric statement:

Corollary 4. *Let $\hat{\Sigma}_d$ be a d-dimensional stratum of a compact Lie group $G \subset$ $\mathrm{GL}_n(\mathbb{R})$. The semialgebraic set $\hat{\Sigma}_d$ is basic open in its Zariski-closure. The topological closure of $\hat{\Sigma}_d$ is a basic closed semialgebraic set in its Zariski-closure. Moreover both sets can be described by at most d strict respectively relaxed inequalities, which is optimal.*

Remark 2. (a) Bröcker and Scheiderer have proved that any basic open set of dimension d can be described by at most d sharp inequalities (unpublished, see Chapter 6.5 in [7]) and that d is also a lower bound. For basic closed sets of dimension d Scheiderer has proved that $\frac{d(d+1)}{2}$ is an upper and lower bound for the number of (relaxed) inequalities required for a description (see [17]). Since Theorem 2.2.3 states that for the (topological) closure of a d-dimensional stratum d inequalities suffice, closures of strata (in particular orbit spaces) form a class of basic closed sets which are easier to describe. Note that the dimension is still a lower bound. Hence there is no gain in efficiency when using generic descriptions.

(b) Suppose that there exist algebraically independent fundamental invariants $\pi_1, \pi_2, \ldots, \pi_m$ of G. If $|G| < \infty$, any d-dimensional stratum can be described by the first d principal minors of $\operatorname{grad}(z)$ (after a permutation of $\pi_1, \pi_2, \ldots, \pi_m$), see [5]. If G is not finite, this is no longer true, see, e.g., Example 3.4.4 or Example 3 in [1].

(c) The upper bound d holds for all d-dimensional basic closed sets, where inequalities are obtained from positive-semidefiniteness conditions on matrices.

3 Constructing the Stratification

As shown in Section 2.2 the d-dimensional components of the strata can be computed by conditions on the rank of the matrix $d\pi(v)$. In this section we provide an algorithm together with necessary tools for the construction of a stratification of the representation- and the orbit space.

More precisely, given a d-dimensional connected component C of a stratum (obtained from rank conditions), the corresponding stratum is given by the orbit of C. The same holds true for the associated semistrata. In this way we construct the stratification of the orbit space out of the stratification of the representation space by computing the image of π (recall Corollary 2.2.1). It remains to add a set of inequalities obtained from the Theorem of Procesi and Schwarz (Theorem 1.2.1), and its refinement (Corollary 2.2.3 and Theorem 2.2.3). We also present an algorithm for computing the stabilizer of a given vector subspace of $\mathbb{K}^n$, which may be used to distinguish the symmetry type of strata[7] of the same dimension.

All used algorithms but the computation of inequalities rely on algebraically closed ground fields. For this reason we present properties of complexifications of real varieties below.

3.1 On the Complexification of a Group-Action

We briefly mention some relations between a compact Lie group G and its complexification and the real- and complex orbit space. More precisely, given fundamental invariants $\pi_1, \pi_2, \ldots, \pi_m \in \mathbb{R}[x_1, x_2, \ldots, x_n]$ of G, in order to describe the orbit space we have to compute the image of the morphism π by Elimination Theory, i.e., one computes the ideal I of relations among $\pi_1, \pi_2, \ldots, \pi_m$, which requires an algebraically closed ground field. As we have already seen, the orbit space of G may be properly be contained in the real algebraic set $\mathcal{V}(I) \subseteq \mathbb{R}^m$. Therefore we have to take care if the computations performed over an algebraically closed field are valid over $\mathbb{R}$. Several important results are based on Kempf-Ness Theory. We refer, e.g., to [19].

Let $G \subset \mathrm{GL}_n(\mathbb{R})$ be a compact Lie group defined by the ideal[8] $I_G \subset \mathbb{R}[s_1, s_2, \ldots, s_m]$. The complexification of G is the zero set of I_G over the complex numbers, denoted by $G_{\mathbb{C}}$. Note that $G_{\mathbb{C}}$ is a complex reductive group with coordinate ring $\mathbb{C}[s_1, s_2, \ldots, s_m]/I_G = \mathbb{R}[s_1, s_2, \ldots, s_m]/I_G \otimes_{\mathbb{R}} \mathbb{C}$ and that G is Zariski-dense in $G_{\mathbb{C}}$. The ideals defining the (real) orbit and the stabilizer of a point $v \in \mathbb{R}^n$ can be computed by Elimination Theory from the ideal I_G and the necessary constructions.

By Hilbert's Finiteness Theorem the invariant ring of G is finitely generated, hence $\mathbb{R}[t_1, t_2, \ldots, t_n]^G = \mathbb{R}[h_1, h_2, \ldots, h_m]$ for some homogeneous invariants $h_1, h_2, \ldots, h_m$. The action of G complexifies to an action of $G_{\mathbb{C}}$ on $\mathbb{C}^n$

[7]Strata of the same dimension may have different stabilizers of the same dimension but different number of connected components

[8]Compact Lie groups are algebraic groups, see for instance [14].

and the invariant ring of $G_{\mathbb{C}}$ equals $\mathbb{C}[t_1, t_2, \dots, t_n]^{G_{\mathbb{C}}} = \mathbb{R}[h_1, h_2, \dots, h_m] \otimes_{\mathbb{R}} \mathbb{C}$. Hence the Hilbert map $\pi : \mathbb{R}^n \to \mathbb{R}^m$ complexifies to $\pi_{\mathbb{C}} : \mathbb{C}^n \to \mathbb{C}^m$ and $\pi_{\mathbb{C}}(\mathbb{C}^n) = \mathrm{cl}_Z(\pi(\mathbb{R}^n))$ (closure in $\mathbb{C}^m$). Let I be the ideal of relations of $h_1, h_2, \dots, h_m$. Since $\mathcal{V}(I) = \mathrm{cl}_Z(\pi(\mathbb{R}^m))$ over $\mathbb{R}$, by Procesi and Schwarz (see Theorem 1.2.1) we have $\mathbb{R}^n/G = \{z \in \mathcal{V}(I) \cap \mathbb{R}^m \mid \mathrm{grad}(z) \geq 0\}$ where the latter closure is taken in $\mathbb{R}^m$.

3.2 Stratification of the Representation Space

By using the results stated in Section 2 we are now able to provide an algorithm for computing a stratification $\Sigma_1, \Sigma_2, \dots, \Sigma_r$ of the representation space of a compact Lie group G. The stratification of the orbit space $\mathbb{R}^m/G$ is obtained by computing the ideals of the images $\pi(\Sigma_1), \pi(\Sigma_2), \dots, \pi(\Sigma_r)$ and adding appropriate inequalities to each set of equations.

```
Algorithm 1 RepSpaceStrata($I_G, \psi$)
In: Ideal defining a compact Lie group $G \subset GLn\mathbb{R}$, $\psi$ a list of polynomials in
$\mathbb{R}[s_1, s_2, \dots, s_k, t_1, t_2, \dots, t_n]$ defining the action of $G$.
Out: list of equations defining the closures $\Sigma_1, \Sigma_2, \dots, \Sigma_r$ of $G$ and their
generic stabilizer .
begin
$\pi = (\pi_1, \pi_2, \dots, \pi_r)$; // algebra generators of $\mathbb{R}[t_1, t_2, \dots, t_n]^G$;
$d = \dim \mathbb{R}^m/G$ // dimension of the orbit space
for $i = 1$ to $d$ do
    $J_d = d \times d$ minors of $d\pi$; // all $d \times d$ minors of the Jacobian
    collectedSpaces = primary decomposition of $\sqrt{J_d}$.
    $c := 1$;
    for each $V \in collectSpaces[i]$ do
        $orbitV = \psi(G, V)$; // orbit of $V$
        if $orbitV \notin \bigcup_{j=1}^{c-1} Semistrata[d][j]$ then begin
            $Semistrata[d][c] = Semistrata[d][c] \cup orbitV$;
            $stabilizer[d][c] = Stabilizer(I_G, \psi, V)$; // representative of the
orbit-type
            $c = c + 1$;
        end
    end-for;
end-for;
return([$Semistrata, stabilizer$]);
end RepSpaceStrata.
```

A set of fundamental invariants for G may be computed by the algorithm given in [6], which works for all reductive groups. Algorithms restricted to compact Lie groups can be found in [9].

We are left with the problem of computing a representative of an orbit type $[v]$, i.e, given the closure $\mathrm{cl}_Z(\Sigma_x)$, find equations for the 'generic' stabilizer G_ξ

of $\mathrm{cl}_Z(\Sigma_x)$. By computing a primary decomposition of the ideal of G_ξ we obtain the index $G_\xi/(G_\xi)_0$

Proposition 11. *Let G be an algebraic group defined by the ideal $I_G \subseteq \mathbb{K}[s_1, s_2, \ldots, s_m]$, let $\alpha : G \times \mathbb{K}^n \to \mathbb{K}^n$ be a linear action, let $V \subseteq \mathbb{K}^n$ be a subspace of dimension d and let $\varphi = (\varphi_1, \varphi_2, \ldots, \varphi_n)$, $\varphi_i \in \mathbb{K}[a_1, a_2, \ldots, a_d]$, be a parametrization of V. Define the ideal $I = \langle I_G, \alpha_i(s,t) - t_i, t_i - \varphi_i : 1 \leq i \leq n\rangle \subset \mathbb{K}(a_1, a_2, \ldots, a_k)[s_1, s_2, \ldots, s_m, t_1, t_2, \ldots, t_n]$ as well as the ideal $J = I \cap \mathbb{K}(a_1, a_2, \ldots, a_k)[s_1, s_2, \ldots, s_m]$ and the (partial) substitution map $\varphi_{\mathbf{b}} : \mathbb{K}(a_1, a_2, \ldots, a_n) \to \mathbb{K}$, $a_i \longmapsto b_i$ for $(b_1, b_2, \ldots, b_n) \in \mathbb{K}^n$. There exists a non-empty Zariski-open set $U \subseteq V$ such that*

$$u \in U \Longrightarrow \varphi_u(J) \cong \mathcal{I}(G_u).$$

Proof. After a finite number of steps we obtain a Gröbner basis of I. In each step we collect the following data: If multiplication by a polynomial f occurs then let P_f be the set of all coefficients of monomials in f which contain some a_i. When computing $f - g$ then add all rational functions in $a_1, a_2, \ldots, a_n$ which are obtained from solving $f - g = 0$ by comparing coefficients. Exclude these sets from $\mathbb{K}^n$. □

Algorithm 2 Stabilizer*(I_G, ψ, I_V)*

In: *ideal I_G of a compact group G, ideal I_V of a component of a stratum.*

Out: *equations of the stabilizer*

Note: *Basering is $\mathbb{K}(a_1, a_2, \ldots, a_k)[s_1, s_2, \ldots, s_k, t_1, t_2, \ldots, t_n]$.*

```
begin
$I$ = GroebnerBasis($I_V$);
$c = 0$;
for $i = 1$ to $n$ do
    if deg(NormalForm($t_i, I$)) $> 0$ then begin
        $c := c + 1$;
        $I$ = GroebnerBasis($I \cup \{t_i - a_c\}$);
    end-if
end-for
$I = I \cup \{\psi_i - t_i : 1 \leq i \leq n\}$;
$J$ = GroebnerBasis($I$) $\cap \mathbb{K}(a_1, a_2, \ldots, a_k)[s_1, s_2, \ldots, s_k]$;
return($J$);
end Stabilizer.
```

Remark 3. An alternative way to compute the number of connected components of the stabilizer is as follows. Compute the generic orbit $G(\xi)$ of V and determine a primary decomposition and the multiplicity of $G(\xi)$ (see [4]).

3.3 Stratification of the Orbit Space

Given a (semi-)stratification of the representation space, the computation of the stratification of the orbit space is essentially the computation of the matrix grad(z) and its symmetric minors. If G is not finite then the dimension of the representation space is strictly greater than the dimension of the orbit space.

The algorithm returns a list of strata of the orbit space of G sorted by dimension. Each stratum $\hat{\Sigma}_{d,i}$ is described as a triple $[[f_1, f_2, \ldots, f_r], [g_1, g_2, \ldots, g_{2^d-1}], [h_1, h_2, \ldots, h_s]]$ where $\hat{\Sigma}_{d,i} = \{z \in \mathbb{R}^m \mid f_1(z) = 0, \ldots, f_r(z) = 0, g_1(z) > 0, \ldots, g_{2^d-1}(z) > 0, h_1(z) \neq 0, \ldots, h_s(z) \neq 0\}$.

```
Algorithm 3 OrbitSpaceStrata($\pi$, repStrata)
In: $\pi = \pi_1, \pi_2, \ldots, \pi_m$ fundamental invariants of $G \subseteq O_{\mathbb{R}}$, list of closures of strata of the representation space. Assume that $d = \dim \mathbb{R}^n/G$.
Out: list of strata of the orbit space (given by equations and inequalities)
begin
grad(z) = $(d\pi_i, d\pi_j)_{i=1..n}^{j=1..n}$;
c = 0;
p(t) = det(grad(z) − t · id_n); // assume $p(z) = t^{m-d}\sum_{i=0}^{d}(-1)^i\delta_i t^i$, characteristic polynomial of grad(z)
for k = 1 to |repStrata| do
    for i = 1 to |repStrata[k]| do
        J = image of repStrata[k][i] under $\pi$. // by Elimination Theory
        ineq = {NormalForm($\delta_i$, J) > 0 | $1 \le i \le d$}
        strata[d][i] = [semistratum[k][i], I, J];
    end-for
end-for
return(strata);
end OrbitSpaceStrata.
```

Remark 4. A stratification up to generic equivalence can be obtained by replacing the line defining I by the line

$I :=$ set of first $d \times d$ principal minors of grad(z); // grad(z) arranged s.t. no principal minor vanishes identically on *repStrata*$[k][i]$.

Example 3. We consider the compact Lie group $G = O_1 \times \mathbf{Z}_2 \subset GL_2(\mathbb{R})$ ($On1$ acting on the first two coordinates, $\mathbf{Z}_2$ acting on the third coordinate) defined by the ideal $\langle s_1^2 + s_2^2 - 1, s_3^2 + s_4^2 - 1, s_1 s_3 + s_2 s_4, s_5^2 - 1\rangle$. The Jacobian of $\pi : \mathbb{R}^3 \to \mathbb{R}^2, (t_1, t_2, t_3) \mapsto (t_1^2 + t_2^2, t_3^2)$ equals $\begin{pmatrix} 2t_1 & 2t_2 & 0 \\ 0 & 0 & 2t_3 \end{pmatrix}$, hence we have (all variables range over $\mathbb{R}$)

$$\begin{aligned} \Sigma_0 &= \{v = (a,b,c) \mid \operatorname{rank} d\pi(v) = 0\} = \{(0,0,0)\} \\ \Sigma_{1,1} \cup \Sigma_{1,2} &= \{v = (a,b,c) \mid \operatorname{rank} d\pi(v) = 1\} = \{(a,b,0) \mid a \neq 0 \text{ or } b \neq 0\} \cup \{(0,0,c) \mid c \neq 0\} \\ \Sigma_2 &= \{v = (a,b,c) \mid \operatorname{rank} d\pi(v) = 2\} = \{(a,b,c) \mid ac \neq 0 \text{ or } bc \neq 0\} \end{aligned}$$

By using the algorithm Stabilizer we obtain for the associated stabilizers the table

Stratum	Σ_0	$\Sigma_{1,1}$	$\Sigma_{1,2}$	Σ_2
Stabilizer	G	$\mathbf{Z}_2 \times \mathbf{Z}_2$	O_1	$\mathbf{Z}_2$

As an example, the ideal $I \subset \mathbb{R}(a_1, a_2)[s_1, s_2, \ldots, s_5]$ defining the generic stabilizer of $\Sigma_{1,1}$ is given by

$$\begin{aligned} I = \langle & a_1 s_3 + a_2 s_4 - a_2, a_1^3 s_2 + a_1 a_2^2 s_3 + a_1^2 a_2 + a_2^3 s_4 - a_1^2 a_2 - a_2^3, \\ & a_1 s_1 + a_2 s_2 - a_1, s_5^2 - 1, a_1^2 + a_2^2 s_4^2 + a_1 a_2 s_3 - a_2^2 s_4 - a_1^2 \rangle \end{aligned}$$

Substitution of $(a, b) \in \Sigma_{1,1}$ for (a_1, a_2) yields the the ideal of the stabilizer of the point (a, b). Inequalities for describing strata of the orbit space are derived from the matrix $\operatorname{grad}(z) = \begin{pmatrix} z_1 & 0 \\ 0 & z_2 \end{pmatrix}$:

$$\begin{aligned} \hat{\Sigma}_0 &= \{(0,0)\} \\ \hat{\Sigma}_{1,1} \cup \Sigma_{1,2} &= \{(z_1, 0) \mid z_1 > 0\} \cup \{(0, z_2) \mid z_2 > 0\} \\ \hat{\Sigma}_2 &= \{(z_1, z_2) \mid z_1 > 0, z_1 z_2 > 0\} \end{aligned}$$

3.4 Examples

Example 4. (See Example 1.2.1) We consider the action of the representation $\mathrm{id} \oplus \mathrm{id}$ on $\mathbb{R}^4$ of $G = O_2 \subset \mathrm{GL}_n(2)\mathbb{R}$, where $\mathrm{id} : G \to \mathrm{GL}_n(2)\mathbb{R}$. Note that the chosen fundamental invariants are algebraically independent. The representation- and orbit space can be decomposed in three strata of dimension $0, 2, 3$ respectively. The matrix $\operatorname{grad}(z)$ is given by $\operatorname{grad}(z) = \begin{pmatrix} 4z_1 & 2z_2 & 0 \\ 2z_2 & z_1 + z_3 & 2z_2 \\ 0 & 2z_2 & 4z_3 \end{pmatrix}$. Strata of the representation space are obtained from rank conditions on $d\pi(x)$.

Dim.	strata on rep. space	strata of orbit space
0	$\Sigma_0 = \{(0,0,0,0)\}$	$\hat{\Sigma}_0 = \{(0,0,0)\}$
2	$\Sigma_2 = \left\{ \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} \lvert t_1 t_4 - t_2 t_3 = 0 \right\} \setminus \Sigma_0$	$\left\{ \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} \in \mathbb{R}^3 \;\middle\vert\; \begin{matrix} z_2^2 - z_1 z_3 = 0 \\ z_1 + z_3 > 0, z_1^2 + 4 z_1 z_3 + z_3^2 > 0 \end{matrix} \right\}$
3	$\Sigma_3 = \mathbb{R}^4 \setminus (\Sigma_0 \cup \Sigma_2)$	$\left\{ \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} \in \mathbb{R}^3 \;\middle\vert\; \begin{matrix} z_1 + z_3 > 0, z_1^2 - 2z_2^2 + 6 z_1 z_3 + z_3^2 > 0, \\ z_1^2 z_3 + z_1 z_3^2 - z^2 (z_1 + z_3) > 0 \end{matrix} \right\}$

Note that the inequality $z_1^2 + 4z_1 z_3 + z_3^2 > 0$ for the description of $\hat{\Sigma}_2$ can be omitted. The inequality $z_1 + z_3 > 0$ cannot be substituted by the principal minors z_1, respectively z_3, which do not vanish identically on $\mathrm{cl}_Z(\hat{\Sigma}_2)$, since such a choice excludes points of the form $z = (0, 0, z_3), z_3 > 0$, respectively $z = (z_1, 0, 0), z_1 > 0$. By using the algorithm Stabilizer we obtain for the associated stabilizers the table

Stratum	Σ_0	Σ_2	Σ_3
Stabilizer	G	$\mathbf{Z}_2$	$\{\mathrm{id}\}$

Example 5. The action of $\mathrm{id} \oplus \mathrm{id}$ of the group $G = SO_2$ (cf. Example 1 in [1]). The polynomials $\pi_1 = t_1^2 + t_2^2 + t_3^2 + t_4^2, \pi_2 = t_1^2 + t_2^2 - t_3^2 - t_4^2, \pi_3 = -2t_1t_4 + 2t_2t_3, \pi_4 = 2t_1t_3 + 2t_2t_4$, as given in [1], form a minimal set of fundamental invariants of $\mathbb{R}[t_1, t_2, \ldots, t_4]^G$. Since $\pi_1, \pi_2, \ldots, \pi_4$ satisfy the relation $\pi_1^2 - \pi_2^2 - \pi_3^2 - \pi_4^2$, the orbit space is embedded in the hypersurface of $\mathbb{R}^4$. There are only two strata of the orbit space. The 4×4 matrix $\mathrm{grad}(z)$ has rank at most 3 and is given by $\mathrm{grad}(z) = \begin{pmatrix} 4z_1 & 4z_2 & 4z_3 & 4z_4 \\ 4z_2 & 4z_1 & 0 & 0 \\ 4z_3 & 0 & 4z_1 & 0 \\ 4z_4 & 0 & 0 & 4z_1 \end{pmatrix}$. We obtain the following description.

Dim.	strata on rep. space	strata of orbit space
0	$\Sigma_0 = \{(0,0,0,0)\}$	$\{(0,0,0,0)\}$
3	$\Sigma_3 = \mathbb{R}^4 \setminus \Sigma_0$	$\left\{ \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} \in \mathbb{R}^4 \;\middle\vert\; \begin{array}{l} z_1^2 - z_2^2 - z_3^2 - z_4^2 = 0 \\ z_4^2 > 0, z_2^2 + z_3^2 + z_4^2 > 0, \\ z_1 z_2^2 + z_1 z_3^2 + z_1 z_4^2 > 0 \end{array} \right\} \subsetneq \mathbb{R}^4.$

The stabilizer associated to Σ_0, respectively, Σ_3 is G, respectively, $\{\mathrm{id}\}$.

Conclusion

We have presented an alternative approach for the computation of stratifications of compact Lie groups and have pointed out, that the dimension of a stratum, respectively, its closure is an upper and lower bound for the number of inequalities, which are necessary in order to describe it. In particular, the number of inequalities for describing orbit spaces is bounded by their dimension. The advantage of the approach lies in the fact, that several applications (like the construction of polynomial potentials) do not necessarily need inequalities at all, and that primary decomposition is faster on the representation space than on the orbit space. Additionally, if the representation of G is not orthogonal, out approach may be used to compute the Zariski-closures of the strata of the orbit space. From a practical point of view, the dependence on orthogonal representations should be avoided.

Acknowledgments

The author is very grateful to Peter Ullrich for numerous useful discussions, particularly about semialgebraic geometry.

References

1. Abud, M., Sartori, G., *The Geometry of Spontaneous Symmetry Breaking.* Ann. Physics, 150 (1983), 307 - 372.

2. Basu, S., Pollack, R., Roy, M.-F. *Algorithms in Real Algebraic Geometry.* Algorithms and Computation in Mathematics , Vol. 10. Springer-Verlag Berlin Heidelberg New York (2003).
3. Bayer, T., `rinvar.lib`.: A SINGULAR 2.0 library for computing invariant rings of reductive groups (2001).
4. Bayer, T. *Computing Stratifications of Quotients of Finite Groups and an Application to Shape Memory Alloys*, pp. 13-24 in: Computer Algebra in Scientific Computing, Proceedings of CASC 01, Editors: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. Springer-Verlag (2001).
5. Bayer, T. *Optimal Descriptions of Orbit Spaces and Strata of Finite Groups*, Preprint, TUM 2004.
6. Bayer, T. *An Algorithm for Computing Invariants of Linear Actions of Algebraic Groups up to a given Degree.* J. Symb. Comp. **35**(4) (2003), pp. 441 - 449.
7. Bochnak, J., Coste, R., Roy, M.-F. *Real Algebraic Geometry.* Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge, Vol. 36. Springer-Verlag Berlin Heidelberg New York (1998).
8. Bredon, G., E. *Introduction to Compact Transformation Groups.* Pure and Applied Mathematics Vol. 46, Academic Press New York London (1972).
9. Gatermann, K., *Computer Algebra Methods for Equivariant Dynamical Systems.* LNM **1728**, Springer-Verlag Berlin Heidelberg New York (2000).
10. Golubitsky, M., Stewart, I., Schaeffer, D. *Singularities and Groups in Bifurcation Theory.* Applied Mathematical Sciences 69, Vol. II, Springer Verlag New York Berlin Heidelberg (1988).
11. Greuel, G.-M., Pfister, G., and Schönemann, H. SINGULAR 2.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern (2001). `http://www.singular.uni-kl.de`.
12. Horn, R.A., Johnson, C.A. *Matrix Analysis.* Cambridge University Press, London New York (1985).
13. Kraft, H., *Geometrische Methoden in der Invariantentheorie.* 2. Auflage, Vieweg Braunschweig-Wiesbaden (1985).
14. Onshchik, A. L., Vinberg, E. B. *Lie Groups and Algebraic Groups.* Springer-Verlag Berlin Heidelberg New York (1990).
15. Procesi, C., Schwarz. G. W. *Inequalities defining orbit spaces.* Invent. Math. **81** (1985), 539 - 554.
16. Rumberger, M. *Eigenvalues on the orbit space.* To appear in: J. Pure and Appl. Alg. **81** (1985), 539 - 554.
17. Scheiderer, C. *Stability index of real varieties.* Invent. Math. **97** (1989), pp. 467-483.
18. Scheja, G., Storch, U., *Lehrbuch der Algebra - Teil 2.* B.G. Teubner, Stuttgart (1988).
19. Schwarz, G. W.*Algebraic Quotients of Compact Group Actions.* J. of Algebra **244** (2001), pp. 365 - 378.
20. Sturmfels, B., *Algorithms in Invariant Theory.* Springer-Verlag Wien New York (1993).
21. Vasconcelos, W., *Computational Methods in Commutative Algebra and Algebraic Geometry.* Algorithms and Computations in Mathematics 2, Springer-Verlag Berlin Heidelberg New York (1998).
22. Zimmer, J., *Mathematische Modellierung und Analyse von Formgedächtnislegierungen in mehreren Raumdimensionen.* PhD. Thesis, Technische Universität München (2000).

SyNRAC: A Maple-package for solving real algebraic constraints — New functions and formula description —

Hitoshi Yanami and Hirokazu Anai

yanami@flab.fujitsu.co.jp
anai@jp.fujitsu.com
[1] Fujitsu Laboratories Ltd.
4-1-1 Kamikodanaka, Nakahara-ku,
Kawasaki, 211-8588, Japan
[2] CREST, Japan Science and Technology Agency
Kawaguchi Center Building, 4-1-8, Honcho, Kawaguchi 332-0012, Japan

Summary. In this paper we present newly developed functions in Maple-package SyNRAC, for solving real algebraic constraints derived from various engineering problems. The current version of SyNRAC provides quantifier elimination (QE) for the quadratic case and a new environment dealing with first-order formulas over the reals (including new simplifiers of formulas) on Maple.

1 Introduction

We presented Maple-package SyNRAC for solving real algebraic constraints in 2003 [4]. SyNRAC stands for a Symbolic-Numeric toolbox for Real Algebraic Constraints and is aimed to be a comprehensive toolbox composed of a collection of symbolic, numeric, and symbolic-numeric solvers for real algebraic constraints derived from various engineering problems.

In this paper we show the current status of development of SyNRAC. In the previous version of SyNRAC [4] the following algorithms were available

- a special QE by the Sturm-Habicht sequence for sign definite condition,
- a special QE by virtual substitution for linear formulas,
- some naive simplifications of quantifier-free formulas.

Besides, the current version of SyNRAC provides the following:

- an environment dealing with first-order formulas over the reals,
- a special QE by virtual substitution for quadratic formulas,
- some new standard simplifiers of formulas.

Since we firstly presented SyNRAC, we have introduced some new operational symbols and fixed a notation system for expressing formulas. We are now developing our tool under the basis of the new environment. The QE algorithms previously equipped have also been reimplemented after the latest setting. These new features greatly extend the applicability and tractability of SyNRAC for solving real algebraic constraints in engineering. The current notation for first-order logic over the reals is much easier to read than the previous one. This helps users describe mathematical formulas for various types of real algebraic constraints. A special QE for quadratic formulas widens the application areas of SyNRAC in actual problems (see [9]). The simplifiers basically reduce the size of a given formula. This contributes not only to improve recognition of formulas but also to remarkably improve the efficiency of special QE procedures based on virtual substitution.

Furthermore, using SyNRAC as a kernel, we are now pushing the further development of design tools based on computer algebra (in particular, QE) in various application fields: One successful attempt is the development of a toolbox for *parametric robust control design* on MATLAB [12] based on the authors' previous works concerning QE-based robust control design [1, 2, 3, 5].

2 A new environment for first-order formulas over the reals

When we say a real algebraic constraint, what we have in mind is a first-order formula over the reals. We describe what type of formulas we are dealing with and how they are expressed in SyNRAC.

An *atomic formula* is an equality or inequality represented by polynomials in a finite number of indeterminates over $\mathbb{Q}$:

$$f(x_1, \ldots, x_n) \ \rho \ g(x_1, \ldots, x_n) \ ,$$

where ρ is one of the relational operators $\{=, \neq, \leq, <\}$. A *formula* is a string obtained by appropriately arranging atomic formulas, logical operators, and existential/universal quantifiers. Here is an example of existential formulas with respect to x, y, and z

$$\exists x \exists y \exists z \ (f_1 \wedge f_2 \wedge (h_1 \vee h_2) \wedge f_3) \Longrightarrow \neg(g_1 \wedge g_2) \ ,$$

where f_i, g_i, and h_i are atomic formulas.

To express formulas in SyNRAC, we need to prepare and fix notational symbols for $\exists$, $\forall$, $\wedge$, $\vee$, $\neg$, and so forth. In the earlier stages of implementation, we were using relational and logical operators bundled in Maple. As we proceeded, it turned out that some of the Maple's operators are unsuitable for our purpose. Let us show a simple example. Let x be just an indeterminate. The `evalb` command, which evaluates a relation in Boolean context, in Maple returns `false` when $x = 0$ is input. This behavior does not meet our

Table 1. The relational operators in SyNRAC

Operator	$=$	$\neq$	$\leq$	$<$	$\geq$	$>$
Notation	`&=`	`&<>`	`&<=`	`&<`	`&>=`	`&>`

Table 2. The logical operators in SyNRAC

Operator	$\wedge$	$\vee$	$\neg$	$\Longrightarrow$	$\Longleftarrow$	$\Longleftrightarrow$
Notation	`&and`	`&or`	`&not`	`&impl`	`&repl`	`&equiv`

Table 3. The quantifiers in SyNRAC

Operator	$\exists x_1 \cdots \exists x_n \varphi$	$\forall x_1 \cdots \forall x_n \varphi$
Notation	`&Ex`$([x_1, \ldots, x_n], \varphi)$	`&All`$([x_1, \ldots, x_n], \varphi)$

expectation, because we want to remain $x = 0$ unchanged unless x is assigned a value.

To avoid such reactions, we have introduced a user-defined operator `&=`[1] and replaced it for the Maple's equality symbol '='. To maintain consistency, the other relational operators are redefined by adding "`&`" at the forefront of the respective commands (see Table 1). Some of them are just an alias for the Maple's corresponding command. Logical operators and quantifier symbols have also been redefined in the same way as in Tables 2 and 3. In SyNRAC, the example formula above is expressed in the following:

`&Ex([x,y,z], (`f_1 `&and` f_2 `&and (`h_1 `&or` h_2`) &and` f_3`) &impl &not(`g_1 `&and` g_2`))` .

The operators `&and` and `&or` can also be used as a prefix operator, taking a list of operands as an argument. The expression

`&and([`f_1`,` f_2`, ...,` f_n`])`

is equivalent in SyNRAC to

f_1 `&and` f_2 `&and` $\cdots$ `&and` f_n .

According to these notational rules, QE algorithms has been (re)implemented in SyNRAC. In addition, several basic utility functions on formulas are provided in SyNRAC, for example, functions for counting the number of atomic formulas, extracting atomic formulas from a formula as a list, and so on. Moreover, some computations for the disjunctive normal form[2] are also available.

[1]A Maple user can form a neutral operator symbol by using `&name` (the ampersand character "`&`" followed by one or more characters).

[2]A formula is called a *disjunctive normal form* if it is a disjunction (a sequence of $\vee$'s) consisting of one or more disjuncts, each of which is a conjunction (a sequence of $\wedge$'s) of one or more atomic formulas.

3 Solving quadratic algebraic constraints over the reals

Here we briefly explain a special QE by virtual substitution of parametric test points that is applicable to formulas in which the quantified variables appear at most quadratically (see [13] for details). We call a formula whose atomic subformulas are at most quadratic (linear) with respect to its quantified variables a *quadratic* (*linear*) formula, respectively.

Let

$$\psi(p_1, \ldots, p_m) \equiv Q_1 x_1 \cdots Q_n x_n \varphi(p_1, \ldots, p_m, x_1, \ldots, x_n)$$

be a linear or quadratic formula, where $Q_i \in \{\forall, \exists\}$ and φ is a quantifier-free formula. By using the equivalence $\forall x \varphi(x) \Longleftrightarrow \neg(\exists x \neg \varphi(x))$, we can change the formula into an equivalent formula of the form $(\neg)\exists x_1 \cdots (\neg)\exists x_n$ $(\neg)\varphi$. The negation '$\neg$' that precedes a quantifier-free formula can be easily eliminated (use De Morgan's law and rewrite the atomic subformulas), which is not essential part of quantifier elimination. Therefore we may focus our attention on an *existential formula,* i.e., a formula of the form $\exists x_1 \cdots \exists x_n \varphi(p_1, \ldots, p_m, x_1, \ldots, x_n)$. Furthermore, it is sufficient to show how to eliminate $\exists x$ in $\exists x \varphi$, since all the quantifiers in the formula can be eliminated by removing one by one from the innermost one.

Now our main purpose is to eliminate the quantified variable $\exists x$ in

$$\psi'(p_1, \ldots, p_m) \equiv \exists x \ \varphi(p_1, \ldots, p_m, x),$$

with $\varphi(p_1, \ldots, p_m, x)$ quantifier-free and quadratic, and obtain an equivalent quantifier-free formula $\psi'(p_1, \ldots, p_m)$. For fixed real values $q_1, \ldots, q_m$ for the parameters $p_1, \ldots, p_m$, all polynomials appearing in $\varphi(x)$ are linear or quadratic. Therefore, the set $M = \{r \in \mathbb{R} | \varphi(q_1, \ldots, q_m, r)\}$ of real values r for x satisfying φ is a finite union of closed, open, or half-open intervals over $\mathbb{R}$. The endpoints of these intervals are among $\pm\infty$ and the real zeros of atomic formulas in φ. Then candidate terms, say, $t_1, \ldots, t_k$, for those zeros can be constructed by the solution formulas for linear or quadratic equations.

If φ does not contain any strict inequalities, all the intervals composing M are either unbounded or closed. In the closed case such an interval contains its real endpoint. So M is nonempty if and only if the substitution of $\pm\infty$ or of one of the candidate solutions t_j for x satisfies φ. Let S be the candidate set $S = \{t_1, \ldots, t_k, \pm\infty\}$. Such a set is called an *elimination set* for $\exists x \varphi$. We obtain a quantifier-free formula equivalent to $\exists x \varphi$ by substituting all candidates in S into φ *disjunctively:*

$$\exists x \varphi \Longleftrightarrow \bigvee_{t \in S} \varphi(x // t).$$

We note that there is a procedure assigning the expression $\varphi(x/t)$ obtained from φ by substituting t for x an equivalent formula [13]. We denote the resulting formula by $\varphi(x//t)$.

If φ contains strict inequalities, we need to add to S other candidates of the form $s \pm \epsilon$, where s is a candidate solution for some left-hand polynomial in a strict inequality and ϵ is a positive infinitesimal.

For improving the efficiency of this method, the following two points are crucial: (i) refining the elimination set S by a scrupulous selection of a smaller number of candidates in S; (ii) integrating with sophisticated simplifications of quantifier-free formulas. SyNRAC now employs three types of elimination sets proposed in [11]. Simplifications in SyNRAC are discussed in the next section.

Moreover, (heuristic) techniques for decreasing the degree during elimination are important for raising the applicability of quadratic QE, because after one quantifier is eliminated for a quadratic case the degree of other quantified variables may increase. Only a simple degree-decreasing functions are implemented in the current version of SyNRAC.

4 Simplification

In the present paper, the term simplification is used for simplification of quantifier-free formulas. When a quantifier is eliminated in a given first-order formula with a special QE procedure, its quantifier-free part usually gets larger. During a QE algorithm, formulas under manipulation tend to get extremely large, deeply nested and highly redundant. That is why simplification procedures, which equivalently change a quantifier-free formula into more concise one, are important in QE. Utilizing simplification algorithms combined with a special QE algorithm contributes to improve not only readability of the resulting formula but efficiency of the computation.

As for simplification, Maple, on which we implement our toolbox SyNRAC, can simplify certain formulas. By using Maple's `evalb` command for the inequality $3 < 5$, the value `true` are obtained. But it does not work for, say, '$x < 3$ `and` $x < 5$'; the `evalb` command does nothing and just returns '$x < 3$ `and` $x < 5$', not '$x < 5$'. Dolzmann and Sturm [8] summarize the rule for simplifying such formulas, to be precise, the formula '$f\ \rho_1\ 0$ `and/or` $g\ \rho_2\ 0$', where f and g differ only by a constant c, and ρ_1 and ρ_2 are an (in)equality. They called these laws *ordering theoretical smart simplification* when $c = 0$, i.e., $f = g$ and *additive smart simplification* when $c \neq 0$, respectively.

Automatic formula simplifiers are implemented in REDLOG[3] and QEPCAD[4] (see [10, 8] for possible simplifications). Several simplification rules including ordering theoretical and additive smart simplification are implemented in SyNRAC, which greatly increase the efficiency of the QE commands in SyNRAC. These rules dramatically work especially when the number of quantified variables are large.

[3]REDLOG is a QE package based on virtual substitution on REDUCE.

[4]QEPCAD is a general QE package that is applicable to all first-order formulas based on cylindrical algebraic decomposition (CAD) [6, 7].

5 Commands in SyNRAC

In this section we show some computational examples to illustrate how commands in SyNRAC are used.[5]
First, you need to load the packages:

```
> read "synrac";   with(combinat);
```

You can use qe_sdc to solve the formula $\forall x > 0,\ f(x) > 0$, called the *sign definite condition* (SDC). The first argument of qe_sdc is polynomial f and the second is the variable to be eliminated. The next example shows how to use the command to solve the problem $\forall x > 0,\ a_2x^2 + a_1x + a_0 > 0$,

```
> qe_sdc(a2*x^2+a1*x+a0, x);

                  ( -a0 &< 0 &and  a1 &< 0 &and -4*a0+a1^2 &< 0 ) &or
                  ( -a0 &< 0 &and -a1 &< 0 &and -4*a0+a1^2 &< 0 ) &or
                  ( -a0 &< 0 &and -a1 &< 0 &and  4*a0-a1^2 &< 0 )

  time = 0.02, bytes = 123614
```

By using qe_lin command, you can solve the existential linear QE problem. This command takes two arguments; the former is a list of quantified variables and the latter a quantifier-free formula. In the following example, qe_lin eliminates the two quantified variables in $\exists x \exists y(y > 2x + 3 \wedge x > 0 \wedge y < s)$ and returns a condition with regard to s.

```
> qe_lin(&Ex([x,y], y&>2*x+3 &and x&>0 &and y&<s));

                                 -1/2*s &< -3/2

  time = 0.03, bytes = 144686
```

The qe_quad command can deal with quadratic QE problems. You can solve the quadratic QE problem $\exists x \exists y(x^2 - 4x - 5 \leq y\ \wedge\ 3 \leq x\ \wedge\ y \leq -5s + 6)$ as follows:

```
> qe_quad(&Ex([x,y], &and[(x^2-4*x-5)&<=y, 3&<=x, y&<=(-5*s+6)]));

                                 -14+5*s &<= 0

  time = 0.03 sec, bytes = 233514
```

The two examples below show that if a decision problem is given, i.e., the input contains no free variables, each command returns the true or false value:

```
> qe_sdc(x^5-x^2+3*x-9,x);
                                    false
  time = 1.11, bytes = 8774262
```

[5]All computations were executed on a Pentium III 1 GHz processor.

```
> qe_lin(&Ex([x,y], y&<2*x+2 and y&<=-3*x+12 and y&>(1/3)*x+5));

                                          true
                     A sample point:  [x, y], [52/25, 144/25]

  time = 0.03, bytes = 155078
```

A sample point is one that makes the formula true.

By calling the qfsimple command, you can simplify quantifier-free formulas with ordering theoretical and additive smart simplification.

```
> qfsimple((x&<5 &and x&<c &and x&>=10) &or (x&<=3 &and x&<=5 &and x&>=-5
           &and x&<>3) &or (x&>7 &and x&<=d));

         (-3+x &<= 0 &and -5-x &<= 0) &or (-x &< -7 &and -d+x &<= 0)

  time = 0.00, bytes = 44974
```

The substsimple command simplifies quantifier-free formulas by making use of simple atomic equations. This command repeats the following two procedures: (i) solving the linear atomic equations with only one variable in each conjunctive formula and substituting its solution for the variable as far as its influence goes; (ii) calling the qfsimple command and simplifying the resulting formula. These are redone until such linear equations run out.

In the next example, z in the input formula is firstly substituted by $3/2$ except the 4th atomic one, and then by using the resulting 1st equation, x is replaced by $3/5$ in three places.

```
> substsimple(5*x&=2*z &and 9&>=3*y-x &and x+4*y+z&>0 &and 2*z-3&=0
             &and 5*x+2*y&<=z+3);

            x &= 3/5 &and -40*y &< 21 &and z &= 3/2 &and -3+4y &<= 0

  time = 0.00, bytes = 97406
```

Example 1. We show an example problem and solve it with SyNRAC. Consider the following convex quadratic programming:

minimize $x_1^2 + x_1x_2 + 2x_2^2$,
subject to $x_1 + 4x_2 \geq 16$, $3x_1 + 2x_2 \geq 18$, $x_1 \geq 0$, $x_2 \geq 0$.

To obtain a description of the first-order formula, we add an unqualified variable z and express the problem in

$$\exists x_1 \exists x_2 (z-(x_1^2+x_1x_2+2x_2^2) \geq 0 \wedge x_1+4x_2 \geq 16 \wedge 3x_1+2x_2 \geq 18 \wedge x_1 \geq 0 \wedge x_2 \geq 0)\,.$$

Eliminating the quantified variables x_1 and x_2, we can obtain a condition on z, from which we would get the range of the objective function. Quantifier elimination procedure in SyNRAC outputs the condition below in 1.78 sec:

```
&or([46 - z &<= 0, &and([567 - 16 z &<= 0,

      &or([(46 - z) &= 0, &and([46 - z &<= 0, -162 + z &<= 0]),

      &and([-466 + z &<= 0, 2659 - 40 z &<= 0]), 2659 - 40 z &<= 0])]),

      &and([46 - z &<= 0, -256 + z &<= 0])]).
```

A little computation tells us that this formula is equivalent to $z \geq 46$. Thus the minimum of the objective function $x_1^2 + x_1x_2 + 2x_2^2$ equals 46.

6 Conclusion

We presented a newly developed functions in Maple-package SyNRAC. The current version of SyNRAC, in particular, provides quantifier elimination for quadratic case and some standard simplifiers of formulas over the new environment for first-order formulas over the reals on Maple. The new features greatly extend the applicability and tractability of SyNRAC for solving real algebraic constraints in engineering. We are continually improving the efficiency of implemented algorithms and are going to implement other algorithms (including symbolic-numeric algorithms) for solving real algebraic constraints into SyNRAC.

Now we note that based on SyNRAC the development of a toolbox for parametric robust control design on MATLAB is ongoing.

We are aware that there is still a considerable way for SyNRAC to be a sophisticated symbolic-numeric tool. Hence we will keep progressing to bridge the gap. Our goal is to develop innovative symbolic-numeric methods and to build novel design tools via SyNRAC for various fields in engineering.

Acknowledgements The authors would like to thank Volker Weispfenning for his invaluable advice.

References

1. H. Anai and S. Hara. Fixed-structure robust controller synthesis based on sign definite condition by a special quantifier elimination. In *Proceedings of American Control Conference 2000*, pages 1312–1316, 2000.
2. H. Anai and S. Hara. Linear programming approach to robust controller design by a quantifier elimination. In *Proceedings of SICE Annual Conference 2002 (Osaka, Japan)*, pages 863–869, 2002.
3. H. Anai and S. Hara. A parameter space approach for fixed-order robust controller synthesis by symbolic computation. In *Proceedings of IFAC World Congress on Automatic Control b'02*, 2002.

4. H. Anai and H. Yanami. SyNRAC: A Maple-package for solving real algebraic constraints. In *Proceedings of International Workshop on Computer Algebra Systems and their Applications (CASA) 2003 (Saint Petersburg, Russian Federation), P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2657*, pages 828–837. Springer, 2003.
5. H. Anai, H. Yanami, and S. Hara. SyNRAC: a maple-package for solving real algebraic constraints toward a robust parametric control toolbox. In *Proceedings of SICE Annual Conference 2003 (Fukui, Japan)*, pages 1716–1721, 2003.
6. G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In H. Brakhage, editor, *Automata Theory and Formal Languages. 2nd GI Conference*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183. Gesellschaft für Informatik, Springer-Verlag, Berlin, Heidelberg, New York, 1975.
7. G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, Sept. 1991.
8. A. Dolzmann and T. Sturm. Simplification of quantifier-free formulae over ordered fields. *Journal of Symbolic Computation*, 24(2):209–231, Aug. 1997.
9. A. Dolzmann, T. Sturm, and V. Weispfenning. Real quantifier elimination in practice. In B. H. Matzat, G.-M. Greuel, and G. Hiss, editors, *Algorithmic Algebra and Number Theory*, pages 221–247. Springer, Berlin, 1998.
10. L. González-Vega. A combinatorial algorithm solving some quantifier elimination problems. In B. Caviness and J. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and monographs in symbolic computation, pages 365–375. Springer-Verlag, 1998.
11. R. Loos and V. Weispfenning. Applying linear quantifier elimination. *The Computer Journal*, 36(5):450–462, 1993. Special issue on computational quantifier elimination.
12. K. Sakabe, H. Yanami, H. Anai, and S. Hara. A MATLAB toolbox for parametric robust control system design based on symbolic computation. In *Bulletin (Kokyuroku) of RIMS (Research Institute for Mathematical Sciences, Kyoto Univ.) Workshop on Computer Algebra—Algorithms, Implementations and Applications 2003 (15-18 December 2003).* To appear.
13. V. Weispfenning. Quantifier elimination for real algebra—the quadratic case and beyond. *Applicable Algebra in Engineering Communication and Computing*, 8(2):85–101, Feb. 1997.

Symbolic-Numeric Sparse Interpolation of Multivariate Polynomials (extended abstract)

Mark Giesbrecht, George Labahn, and Wen-shin Lee

[1] School of Computer Science, University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
[2] INRIA, GALAAD
BP 93, 06902 Sophia-Antipolis, France

1 Introduction

In many applications, multivariate polynomials are encountered in such a way that an implicit representation is the most cost effective for computation. A black-box representation of a multivariate polynomial is a procedure such that, for any given input, it outputs the evaluation of the polynomial at that input. Black-box polynomials appear naturally in applications such as multivariate polynomial systems, both approximate and exact [Corless et al.(2001)Corless, Giesbrecht, Kotsireas, and Watt], and the manipulation of sparse polynomials, such as factoring [Kaltofen and Trager(1990), Díaz and Kaltofen(1998)].

We consider the problem of sparse interpolation of a multivariate polynomial given by a floating-point black box. That is, both the inputs and outputs of the black box are precise up to a fixed number of digits. As a result, the coefficients in the target polynomial can only be known up to a certain precision.

For example, a floating-point black box can be a procedure that, for any given input, computes the value of the determinant of a matrix of multivariate polynomials with floating-point coefficients. Such a black-box evaluation procedure could be constructed from a number of effective numeric algorithms for finding a determinant, for example Gaussian elimination.

A typical problem with black-box representations is to convert such objects into a standard representation. That is, given a multivariate polynomial $f(x_1, \ldots, x_n)$ in black-box form, we want to find powers $(d_{j_1}, \ldots, d_{j_n})$ and non-zero coefficients c_j such that

$$f(x_1, \cdots, x_n) = \sum_{j=1}^{t} c_j x_1^{d_{j_1}} \cdots x_n^{d_{j_n}}.$$

In the case of multivariate polynomials, one often expects such a representation to be sparse. The best known interpolation methods that are sensitive to the sparsity of the target polynomial are the Ben-Or/Tiwari algorithm [Ben-Or and Tiwari(1988)] and Zippel's method [Zippel(1979)]. Although both approaches have been generalized and improved [Zippel(1990), Kaltofen and Lakshman Yagati(1988), Grigoriev et al.(1990)Grigoriev, Karpinski, and Singer, Zilic and Radecka(1999)], they all require exact arithmetic. In a different model, Mansour [Mansour(1995)] gives an impressive randomized algorithm for interpolating a sparse integer polynomial from (limited precision) interpolation points. While the algorithm guarantees an answer with controllably high probability, its cost is quite dependent on the size L (regardless of its precision) of the largest coefficient in f, as well as the sparsity t and degree: it requires about $O((\log L)^8 t \log \deg f)$ bit operations.

In numeric arithmetic, a procedure comparable to Ben-Or/Tiwari algorithm dates back to Baron de Prony in 1795 [Prony(1795), Brezinski(1991)], which actually considers the interpolation problem of fitting a sum of univariate exponential functions:

$$F(x) = \sum_{j=1}^{t} c_j e^{\mu_j x}.$$

Prony's method determines c_j and μ_j from the evaluations of $F(x)$ at equally spaced points $F(0), F(1), \ldots$. There are many interesting variations of this numeric interpolation problem, for example, the problem of shape from moments [Milanfar et al.(1995)Milanfar, Verghese, Karl, and Wilsky, Golub et al.(1999)Golub, Milanfar, and Varah]. Some other examples are tomography [Milanfar et al.(1995)Milanfar, Verghese, Karl, and Wilsky] and signal decomposition [Marple, Jr.(1987)].

We develop sparse interpolation algorithms for multivariate polynomials in floating-point arithmetic. We also take advantage of current state-of-the-art algorithms and look at the accuracy for numeric subproblems. Finally, we conclude with a discussion of relavant topics and future research.

2 Preliminaries

In this section we describe Prony's method for interpolating sums of exponentials and the Ben-Or/Tiwari algorithm for multivariate polynomials. We show that these two algorithms are closely related.

2.1 Prony's method

Prony's method [Prony(1795)] seeks to interpolate a univariate $F(x)$ that is a sum of exponential functions. That is, it tries to determine c_j and μ_j such that

$$F(x) = \sum_{j=1}^{t} c_j e^{\mu_j x} \text{ with } c_j \neq 0.$$

Since there are $2t$ unknowns, one would expect a system of at least the same number of equations. However, these equations are not linear. Prony's method converts the nonlinear component to the root finding of a single, univariate polynomial. All other steps involve solving systems of (structured) linear equations.

Let $b_j = e^{\mu_j}$, then $F(x) = \sum_{j=1}^{t} c_j e^{\mu_j x} = \sum_{j=1}^{t} c_j b_j^x$. Consider the polynomial $\Lambda_F(z)$ having the b_j's as zeros:

$$\Lambda_F(z) = \prod_{j=1}^{t} (z - b_j) = z^t + \lambda_{t-1} z^{t-1} + \cdots \lambda_1 z + \lambda_0.$$

The key facts used in the algorithm are that the sequence $F(0), F(1), F(2), \ldots$ is linearly generated, and its minimal generating polynomial is Λ_F [Hildebrand(1956), Ben-Or and Tiwari(1988)].

The polynomial $\Lambda_F(z)$ can be determined by solving a Hankel system:

$$\begin{bmatrix} F(0) & F(1) \ldots & F(t-1) \\ F(1) & F(2) \ldots & F(t) \\ \vdots & \vdots \ \ddots & \vdots \\ F(t-1) & F(t) \ \ldots & F(2t-2) \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \vdots \\ \lambda_{t-2} \\ \lambda_{t-1} \end{bmatrix} = - \begin{bmatrix} F(t) \\ F(t+1) \\ \vdots \\ F(2t-1) \end{bmatrix}, \tag{1}$$

which is also called the *Yule-Walker* equations for the series $\sum_{j \geq 0} F(j) z^j$.

Finding zeros for Λ_F can determine $b_1, \ldots, b_t$ (thus $\mu_1, \ldots, \mu_t$). The coefficients $c_1, \ldots, c_t$ can be computed by solving a transposed Vandermonde system:

$$\begin{bmatrix} 1 & \cdots & 1 \\ b_1 & \cdots & b_t \\ \vdots & \ddots & \vdots \\ b_1^{t-1} & \cdots & b_t^{t-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_t \end{bmatrix} = \begin{bmatrix} F(0) \\ F(1) \\ \vdots \\ F(t-1) \end{bmatrix} \tag{2}$$

2.2 The Ben-Or/Tiwari method

For a given black-box polynomial f with n variables, the Ben-Or/Tiwari method [Ben-Or and Tiwari(1988)] finds coefficients c_j and integer exponents $(d_{j_1}, \ldots, d_{j_n})$ such that

$$f(x_1, \ldots, x_n) = \sum_{j=1}^{t} c_j x_1^{d_{j_1}} \cdots x_n^{d_{j_n}} \text{ with } c_j \neq 0.$$

Let $\beta_j(x_1, \ldots, x_n) = x_1^{d_{j_1}} \cdots x_n^{d_{j_n}}$ be the j-th term in f, and set $b_j = \beta_j(p_1, \ldots, p_n) = p_1^{d_{j_1}} \cdots p_n^{d_{j_n}}$ with $p_1, \ldots, p_n$ pairwise relatively prime. Note that $b_j^k = \beta_j(p_1^k, \ldots, p_n^k)$ for any power k.

Then we consider the function $F(k) = f(p_1^k, \ldots, p_n^k)$. Designed for exact arithmetic, the Ben-Or/Tiwari algorithm solves Yule-Walker equations in (1) by the Berlekamp/Massey algorithm from coding theory. Once the terms b_j are found through the root finding of $\Lambda_F(z) = 0$, the exponents $(d_{j_1}, \ldots, d_{j_n})$ can be determined via repeatedly dividing b_j by $p_1, \ldots, p_n$ that are relatively prime. Finally, the coefficients c_j are determined by solving a Vandermonde system similar to (2).

3 Numerical methods in sparse interpolations

We give two sparse algorithms for interpolating black-box multivariate polynomials in floating-point arithmetic. One basically follows the steps of the Ben-Or/Tiwari algorithm [Ben-Or and Tiwari(1988)], while the other takes advantage of the generalized eigenvalue reformulation of Prony's method [Golub et al.(1999)Golub, Milanfar, and Varah]. We also look at the stability of the main subproblems in these algorithms.

3.1 Ben-Or/Tiwari algorithm in floating-point arithmetic

If the steps of the Ben-Or/Tiwari algorithm are directly executed in floating-point arithmetic, severe difficulties arise at various stages of the computation. First, to solve the Yule-Walker equations in (1), rather than using the Berlekamp/Massey algorithm in exact arithmetic, we need to solve a Hankel system that is often ill-conditioned, especially in the case of real numbers [Beckermann(2000)]. Even worse, the solutions λ_j to such Hankel system are coefficients of the generating polynomial $\Lambda(z) = z^t + \lambda_{t-1} z^{t-1} + \cdots + \lambda_1 z + \lambda_0$ for root finding, while root finding is usually very sensitive to the perturbations in λ_j. Finally, the coefficients c_j of our target polynomial are determined by solving a Vandermonde system, which might be ill-conditioned.

The above difficulties are also shared by Prony's method for interpolating a sum of exponential functions, even though it was designed and used as a numerical method. Special challenges also exist for multivariate polynomial interpolations. For example, unlike the case in exact arithmetic, we can no longer use integer factorizations to recover the exponent of each variable in a multivariate term.

By evaluating each variable at powers of an appropriate primitive root of unity, we can improve the conditioning of the associated Hankel system, the computation of zeros for the generating polynomial, and the Vandermonde system. Furthermore, the exponent of each variable in a multivariate term can also be recovered.

Our strategy is to evaluate at powers of primitive roots of unity whose orders are pairwise relatively prime. Let $f(x_1, \ldots, x_n) = \sum_{j=1}^t c_j x_1^{d_{j_1}} \cdots x_n^{d_{j_n}} = \sum_{j=1}^t c_j \beta_j(x_1, \ldots, x_n)$ with $c_j \neq 0$. If $p_1, \ldots, p_n \in \mathbb{Z}_{>0}$ are pairwise relatively prime and $p_k > \deg_{x_k} f$ for $1 \le k \le n$ (assume we have an upper degree bound for every variable in f). Consider the following sequence for interpolation:

$$\alpha_s = f(\omega_1^s, \omega_2^s, \ldots, \omega_n^s) \text{ for } 0 \le s \le 2t-1$$

with $\omega_k = \exp(2\pi i/p_k)$. Set $m = p_1 \cdots p_n$ and $\omega = \exp(2\pi i/m)$, then $\omega_k = \omega^{m/p_k}$ for $1 \le k \le n$.

In $f(\omega_1, \ldots, \omega_n)$, each term $\beta_j(x_1, \ldots, x_n)$ is mapped to value $\beta_j(\omega_1, \ldots, \omega_n) = \omega^{d_j}$. In a numeric setting, each d_j can be computed by rounding $\log_\omega(\beta_j(\omega_1, \ldots, \omega_n))$ to the nearest integer. Then the exponent for each variable $(d_{j_1}, \ldots, d_{j_n}) \in \mathbb{Z}^n_{>0}$ can be uniquely determined by the reverse steps of the Chinese remainder algorithm (cf. [Geddes et al.(1992)Geddes, Czapor, and Labahn]). That is, $d_j \bmod p_k \equiv d_{j_k}$ for $1 \le k \le n$, and[3]

$$d_j = d_{j_1} \cdot \left(\frac{m}{p_1}\right) + \cdots + d_{j_n} \cdot \left(\frac{m}{p_n}\right). \tag{3}$$

In the remainder of this subsection, we look at the numerical sensitivity of solving the associated Hankel system and the root finding of generating polynomial $\Lambda(z)$. The separation of powers for polynomial terms and the solving of the associated Vandermonde system will be addressed in subsections 3.3 and 3.4.

Solving the associated Hankel system

Consider polynomial $f(x_1, \ldots, x_n) = \sum_{j=1}^t c_j x_1^{d_{j_1}} \cdots x_n^{d_{j_n}} = \sum_{j=1}^t c_j \beta_j(x_1, \ldots, x_n)$ and the evaluation sequence $\alpha_s = f(a_1^s, \ldots, a_n^s)$ for $0 \le s \le 2t-1$. We need to solve the following Hankel systems $H_{0,t-1}$:

$$\underbrace{\begin{bmatrix} \alpha_0 & \alpha_1 & \ldots & \alpha_{t-1} \\ \alpha_1 & \alpha_2 & \ldots & \alpha_t \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{t-1} & \alpha_t & \ldots & \alpha_{2t-2} \end{bmatrix}}_{H_{0,t-1}} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{t-1} \end{bmatrix} = - \begin{bmatrix} \alpha_t \\ \alpha_{t+1} \\ \vdots \\ \lambda_{2t-1} \end{bmatrix}$$

In general, if a polynomial f is evaluated at powers of a real value, the difference between the scale of varying powers contributes to the ill conditioning of the Hankel system. This problem is avoided in our method, since our $H_{0,t-1}$ is formed from the evaluations on a unit circle.

Now let $b_j = \beta_j(\omega_1, \ldots, \omega_n)$, $D = \mathrm{diag}(c_1, \ldots, c_t)$, and

$$V = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ b_1 & b_2 & \ldots & b_t \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{t-1} & b_2^{t-1} & \ldots & b_t^{t-1} \end{bmatrix}.$$

[3]Recall that in exact arithmetic, the original Ben-Or/Tiwari algorithm evaluates variables at values that are pairwise relatively prime.

The Hankel system $H_{0,t-1}$ can be factored as $H_{0,t-1} = VDV^{\mathrm{Tr}}$. We make use of this factorization for investigating the condition of $H_{0,t-1}$.

Lemma 1.

$$\|H_{0,t-1}^{-1}\| \geq \frac{1}{t} \max_j \frac{1}{|c_j|}.$$

Proof. Let D_j be the matrix derived from D by using 0 to replace c_j in the diagonal, then matrix VD_jV^{Tr} is singular for $1 \leq j \leq t$.

Using the Eckart-Young theorem and the fact that the b_j's are on the unit circle,

$$\begin{aligned}\frac{1}{\|H_{0,t-1}^{-1}\|} &= \min\{\|H_{0,t-1} - A\|,\ A \text{ singular }\}\\ &\leq \min\{\|H_{0,t-1} - VD_jV^{\mathrm{Tr}}\|\}\\ &\leq \|[1, b_j, \ldots, b_j^{t-1}]\|^2 \cdot |c_j| \leq t \cdot |c_j|.\end{aligned}$$

Lemma 2.

$$\|H_{0,t-1}^{-1}\| \leq \|V^{-1}\|^2 \cdot t \cdot \max_j \frac{1}{|c_j|}.$$

Proof. Consider $H_{0,t-1}^{-1} = (V^{\mathrm{Tr}})^{-1}D^{-1}V^{-1}$, then

$$\begin{aligned}\|H_{0,t-1}^{-1}\| \leq \|V^{-1}\|^2\|D^{-1}\| &\leq \|V^{-1}\|^2 \cdot \sum_{j=1}^{t} \|D^{-1}e_j\|\\ &\leq \|V^{-1}\|^2 \cdot t \cdot \max_j \frac{1}{|c_j|}.\end{aligned}$$

An upper bound for $\|H_{0,t-1}^{-1}\|$ involves the conditioning of the Vandermonde system V, which will be discussed in subsection 3.4.

Root finding of the generating polynomial

The solutions of the associated Hankel system λ_j are coefficients in the polynomial $\Lambda(z) = z^t + \lambda_{t-1}z^{t-1} + \cdots + \lambda_1 z + \lambda_0$. In our floating-point Ben-Or/Tiwari interpolation steps, the zeros of $\Lambda(z)$ are $b_j = \beta_j(\omega_1, \ldots, \omega_n)$, where $f(x_1, \ldots, x_n) = \sum_{j=1}^{t} c_j x_1^{d_{j1}} \cdots x_n^{d_{jn}} = \sum_{j=1}^{t} c_j \beta_j(x_1, \ldots, x_n)$, and b_j are on the unit circle.

It is well known that root finding for a polynomial is generally poorly conditioned with respect to perturbations in the coefficients [Wilkinson(1963)]. However, the conditioning is greatly improved when all the roots are on the unit circle.

Let b_k be a zero of $\Lambda(z)$ and $\tilde{b}_k$ a zero of $\Lambda(z) + \epsilon\Gamma(z)$, where

$$\Gamma(z) = \gamma_t z^t + \gamma_{t-1}z^{t-1} + \cdots \gamma_0,$$

then $\tilde{b}_k \approx b_k + \zeta_1\epsilon$ and $\Lambda(b_k + \zeta_1\epsilon) + \epsilon\Gamma(b_k + \zeta_1\epsilon) \approx 0$. By a Taylor expansion with respect to b_k, we see

$$\sum_{j=0}^{t} \frac{1}{j!} \Lambda^{(j)}(b_k) \cdot (\zeta_1 \epsilon)^j + \epsilon \sum_{j=0}^{t} \frac{1}{j!} \Gamma^{(j)}(b_k) \cdot (\zeta_1 \epsilon)^j \approx 0.$$

Recall that $\Lambda(b_k) = 0$, and consider only the first order terms in ϵ, we have $\Lambda^{(1)}(b_k) \cdot \zeta_1 \epsilon + \epsilon \Gamma(b_k) \approx 0$ and

$$|\zeta_1| \approx \left| \frac{\Gamma(b_k)}{\Lambda^{(1)}(b_k)} \right| \leq \frac{\sum_{j=0}^{t} |\gamma_j|}{|\prod_{j \neq k}(b_k - b_j)|}.$$

Therefore,

$$|b_k - \tilde{b}_k| < \epsilon \cdot \frac{K_1}{|\prod_{j \neq k}(b_k - b_j)|} + K_2 \epsilon^2.$$

The size of $|\prod_{j \neq k}(b_k - b_j)|$ depends on the distribution of b_j's on the unit circle (cf. subsection 3.4).

3.2 Generalized eigenvalue reformulation

We present an alternative interpolation algorithm by adapting the reformulation of Prony's method that combines the solving of a Hankel system and the root finding of a polynomial into a single generalized eigenvalue problem []GMV99. As a result, both solving the Hankel system and finding roots for a polynomial can be avoided.

Consider the polynomial $f(x_1, \ldots, x_n) = \sum_{j=1}^{t} c_j x_1^{d_{j1}} \cdots x_n^{d_{jn}} = \sum_{j=1}^{t} c_j \beta_j(x_1, \ldots, x_n)$ and the evaluation sequence $\alpha_s = f(a_1^s, \ldots, a_n^s)$ for $0 \leq s \leq 2t-1$. Define Hankel systems

$$H_{0,t-1} = \begin{bmatrix} \alpha_0 & \ldots & \alpha_{t-1} \\ \vdots & \ddots & \vdots \\ \alpha_{t-1} & \ldots & \alpha_{2t-2} \end{bmatrix} \text{ and } H_{1,t} = \begin{bmatrix} \alpha_1 & \ldots & \alpha_t \\ \vdots & \ddots & \vdots \\ \alpha_t & \ldots & \alpha_{2t-1} \end{bmatrix},$$

so that $H_{1,t}$ is one row shifted up from $H_{0,t-1}$. Let $b_j = \beta_j(a_1, \ldots, a_n)$. If we set $Z = \mathrm{diag}(b_1, \ldots, b_t)$, $D = \mathrm{diag}(c_1, \ldots, c_t)$, and

$$V = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ b_1 & b_2 & \ldots & b_t \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{t-1} & b_2^{t-1} & \ldots & b_t^{t-1} \end{bmatrix},$$

then $H_{0,t-1} = VDV^{\mathrm{Tr}}$, $H_{1,t} = VDZV^{\mathrm{Tr}}$. The solutions λ to the generalized eigenvalue problem

$$H_{1,t} v = \lambda H_{0,t-1} v \tag{4}$$

are $b_1, \ldots, b_t$, the terms $\beta_j(x_1, \ldots, x_n)$ evaluated at $(a_1, \ldots, a_n)$. If $a_1, \ldots, a_n$ are chosen as $\omega_1, \ldots, \omega_n$ in subsection 3.1, the multivariate terms $\beta_j(x_1, \ldots, x_n)$ can be recovered accordingly.

[Golub et al.(1999)Golub, Milanfar, and Varah] show that the generalized eigenvalue problem

$$(H_1 - \lambda H_0)v = 0 \tag{5}$$

can be analyzed for errors to the first order. For a given eigenvalue λ and the associated eigenvector v, suppose

$$(H_1 + \epsilon \hat{H}_1)(v + \epsilon v^{(1)} + \cdots) = (\lambda + \epsilon\lambda^{(1)} + \cdots)(H_0 + \epsilon\hat{H}_0)(v + \epsilon v^{(1)} + \cdots)$$

is an ϵ-perturbation of our eigenvalue problem. Looking only at first order errors gives

$$(H_1 - \lambda H_0)v^{(1)} = (\lambda^{(1)} H_0 + \lambda\hat{H}_0 - \hat{H}_1)v. \tag{6}$$

Since v is both a left and right eigenvector (both H_0 and H_1 are symmetric), the left side of (6) is annihilated by multiplication on the left by v^{Tr}:

$$\lambda^{(1)} = \frac{v^{\mathrm{Tr}}(\hat{H}_1 - \lambda\hat{H}_0)v}{v^{\mathrm{Tr}} H_0 v}. \tag{7}$$

Assume the perturbations are of the same size as the precise value, that is, $\|\hat{H}_0\|_2 = \|H_0\|_2$ and $\|\hat{H}_1\|_2 = \|H_1\|_2$, and v is normalized as a unit vector, then (7) gives the error bound

$$\|\lambda^{(1)}\| \le \frac{\|H_1\|_2 + \|\lambda\| \|H_0\|_2}{|v^{\mathrm{Tr}} H_0 v|}.$$

Assuming $\|H_1\|_2 = \|H_0\|_2$ (which is reasonable since the matrices have such close structures) and recalling λ is a root of unity so that $\|\lambda\| = 1$ give

$$\|\lambda^{(1)}\| \le \frac{2\|H_0\|_2}{|v^{\mathrm{Tr}} H_0 v|}. \tag{8}$$

Since the norm of H_0 can always be bounded by scaling when necessary, the interesting quantity in the error bound (8) is

$$\frac{1}{|v^{\mathrm{Tr}} H_0 v|}. \tag{9}$$

The coefficients c_j's play a role in bounding the errors of the generalized eigenvalues. Notice that the columns of $(V^T)^{-1}$ give both the right and left eigenvectors of (5). If λ_j is the eigenvalue corresponding with the j-th column of $(V^T)^{-1}$, that is $v_j = (V^T)^{-1} e_j$ for (5), then (9) can be reduced to

$$\frac{|v_j^{\mathrm{Tr}} \cdot v_j|^2}{|v_j^{\mathrm{Tr}} H_0 v_j|} = \frac{\|v_j\|^2}{|v_j^{\mathrm{Tr}} \cdot V \cdot D \cdot V^{\mathrm{Tr}} \cdot v_j|} = \frac{\|(V^T)^{-1} e_j\|^2}{|c_j|} \le \frac{\|(V^T)^{-1}\|^2}{|c_j|}.$$

An eigenvalue is *well-disposed* [Golub et al.(1999)Golub, Milanfar, and Varah] if the quantity (9) is "small". From (8) we see that a well-disposed eigenvalue will result in a small error in the computation of the associated eigenvalue. The advantage of solving the generalized eigenvalue problem in (4) is that there are stable numerical methods for such a problem and those methods do not require $b_1, \ldots, b_n$ to be on a unit circle.

3.3 Separation of powers

After determining the term values b_j's, we still needs to consider the precision required for correctly recovering the integer exponents through taking the logarithms of $b_j = \omega^{d_j}$ with $\omega = \exp(2\pi i/m)$.

Since two consecutive m-th roots of unity on the unit circle are separated by an angle of radian $\frac{2\pi}{m}$, the distance between these two points is bounded below by twice the sine of half the angle between them. Thus, in order to separate any two such points by rounding one must have values correct to

$$\frac{1}{2}|2\sin(\frac{\pi}{m})| \approx \frac{\pi}{m} \text{ and } m = p_1 \cdots p_n$$

with p_k primes and $p_k > \deg f_{x_k}$.

3.4 Recovering the coefficients

Once the term values b_j's have been determined, it still remains to compute their coefficients c_j's, which can be done in a number of ways [Golub et al.(1999)Golub, Milanfar, and Varah]. If the term values are determined as general eigenvalues in (5) by the QZ algorithm, the computed eigenvectors v_j can be put to use here. Let M be the matrix whose columns are eigenvectors v_j, then the coefficients can be computed by

$$c_j = (v_j^T H_0 v_j)(T_{j,1})^2,$$

with $T = M^{-1} = S^{-1}V^T$ and S a diagonal scaling matrix [Golub et al.(1999)Golub, Milanfar, and Varah]. Alternatively, we can directly solve the Vandermonde system (2) to determine the c_j's.

The conditioning of the associated Vandermonde system

While Vandermonde matrices can have poor conditioning, especially over the reals [Beckermann(2000), Gautschi and Inglese(1988)], our problem is better behaved because all our points lie on the unit circle. For example, for a $t \times t$ Vandermonde matrix, if the nodes happen to be all the t-th roots of unity, the condition number for the 2-norm is 1, which is optimal [Gautschi(1975), Example 6.4]. A Vandermonde matrix can be well conditioned even if it is roots of unity which are not perfectly uniform. One particular distribution of roots of unity has been studied in [Córdova et al.(1990)Córdova, Gautschi, and Ruscheweyh]. They consider a *Van der Corput sequence*, that is, $\{a_j\}_{j=0}^{\infty}$ for

$$a_j = \sum_{k=0}^{\infty} j_k 2^{-(k+1)} \text{ with } j = \sum_{k=0}^{\infty} j_k 2^k \text{ for } j_k \in \{0, 1\}.$$

For a $t \times t$ Vandermonde matrix with nodes $\exp(2\pi i a_0)$, $\ldots$, $\exp(2\pi i a_{t-1})$, the 2-norm condition number is less than $\sqrt{2t}$ [Córdova et al.(1990)Córdova, Gautschi, and Ruscheweyh, Corollary 3].

In general, the conditioning of a Vandermonde matrix depends on the distance between different nodes, which follows from a well-known explicit formula for the inverse. If V is a $t \times t$ Vandermonde matrix with nodes $z_1, \ldots, z_t$, then $V^{-1} = [a_{j,k}]$ where

$$\ell_j(z) = a_{j,1} + a_{j,2}z + \cdots + a_{j,t}z^{t-1} = \prod_{\substack{u=1 \\ u \neq j}}^{t} \frac{z - z_u}{z_j - z_u}$$

is the j-th Lagrange polynomial of points $z_1, \ldots, z_t$. In our case, $z_1, \ldots, z_t$ are all on the unit circle. We have the following upper and lower bounds for $\|V^{-1}\|$ in the infinity norm [Gautschi(1975)]:

$$\max_{1 \le j \le t} \prod_{\substack{u=1 \\ u \neq j}}^{t} \frac{1}{|z_j - z_u|} < \|V^{-1}\|_\infty \le \max_{1 \le j \le t} \prod_{\substack{u=1 \\ u \neq j}}^{t} \frac{2}{|z_j - z_u|}.$$

Now we look at the condition of our associated Vandermonde system. Suppose $p_1, \ldots, p_n$ are distinct primes, $p_k > \deg_{x_k} f$, and $\omega = \exp(2\pi i/m)$ for $m = p_1 \cdots p_n$. If the target polynomial f is evaluated at powers of $(\omega_1, \ldots, \omega_n)$ for $\omega_k = \omega^{m/p_k}$, the distribution of term values on the unit circle is fixed because the polynomial terms are fixed.

To scrutinize the distribution of term values, instead of $f = \sum_{j=1}^{t} c_j x_1^{d_{j_1}} \cdots x_n^{d_{j_n}}$, we consider the univariate $f_1(x) = \sum_{j=1}^{t} c_j x^{d_j}$ with $d_j = d_{j_1} \cdot (m/p_1) + \cdots + d_{j_n} \cdot (m/p_n)$. Now the term values are $\omega^{d_1}, \ldots, \omega^{d_n}$, and their distribution on the unit circle depends on m and the differences between exponents d_j and d_k for $j \neq k$. If some of them are bunched together, then comparing to m some d_j's are relatively close to each other, which may lead to the ill-conditioning of the Vandermonde system.

Therefore, we propose to randomize the distribution of term values on the unit circle. Under the condition that $p_1, \ldots, p_n$ are given (hence their product m), we randomly pick an integer r such that $0 < r < \min(p_1, \ldots, p_n)$ and consider f evaluated at powers of $(\omega_1^r, \ldots, \omega_n^r)$. Now the corresponding univariate f_1 are evaluated at powers of ω^r, but it can also be viewed as another univariate polynomial $f_1^{[r]} = \sum_{j=1}^{t} c_j x^{d_j \cdot r}$ evaluated at powers of ω. In other word, the difference between ever pair of exponents is now multiplied by a random r in mod m, and may produce a distribution of term values that provides a better condition for the Vandermonde system. (As for the interpolation result, the original polynomial terms can be recovered by dividing each exponent by r in mod m.)

We implement the randomization strategy as the following: the user determines an integer threshold $\zeta > 0$ and randomly picks ζ distinct integers $r_1, \ldots, r_\zeta$ such that $0 < r_k < \min(p_1, \ldots, p_n)$ for each interpolation attempt. It is very likely that at least one of these ζ interpolation results is based on the computation of a "well-distributed" term values. If $\zeta \ge 2$, we may check whether there are two (or more) interpolation results that are very closed to each other. Such results are likely to be good approximations of the target polynomial.

We can further randomize the multiples of differences for exponents in each variable. For each interpolation attempt, we pick n random integers $r_1, \ldots, r_n$ such that $0 < r_k < p_k$ for $1 \le k \le n$, then interpolate the polynomial f evaluated at the powers of $(\omega_1^{r_1}, \ldots, \omega_n^{r_n})$. Rather than being multiplied by a random r in mod m, the exponent difference $d_j - d_k$ becomes a sum of random multiples of its components in mod m. That is,

$$r_1 \cdot (d_{j_1} - d_{k_1}) \cdot \left(\frac{m}{p_1}\right) + \cdots + r_n \cdot (d_{j_n} - d_{k_n}) \cdot \left(\frac{m}{p_n}\right).$$

Similarly, to recover the original terms in the interpolation result, the exponents in each variable x_k are divided by r_k accordingly.

4 Future directions

We have implemented our methods in Maple and are currently conducting experiments. A full sensitivity analysis, to obtain stronger guarantees, is being pursued. Furthermore, the generalized eigenvalue approach may be exploited for interpolation at real values.

On the other hand, based on the polynomial relations between trigonometric functions, we have extended our sparse interpolation methods to the interpolation of trigonometric functions. We intend to further investigate the sensitivity and experiments in this context as well.

Acknowledgements

We thank Annie Cuyt and Brigitte Verdonk for pointing recent related works. We also thank Bernhard Beckermann and Erich Kaltofen for their comments.

References

[Beckermann(2000)] Beckermann, B., 2000. The condition number of real Vandermonde, Krylov and positive definite Hankel matrices. Numeriche Mathematik 85, 553–577.

[Ben-Or and Tiwari(1988)] Ben-Or, M., Tiwari, P., 1988. A deterministic algorithm for sparse multivariate polynomial interpolation. In: Proc. Twentieth Annual ACM Symp. Theory Comput. ACM Press, New York, N.Y., pp. 301–309.

[Brezinski(1991)] Brezinski, C., 1991. History of Continued Fractions and Padé Approximants. Springer Verlag, Heidelberg, Germany.

[Córdova et al.(1990)Córdova, Gautschi, and Ruscheweyh] Córdova, A., Gautschi, W., Ruscheweyh, S., 1990. Vandermonde matrices on the circle: spectral properties and conditioning. Numerische Mathematik 57, 577–591.

[Corless et al.(2001)Corless, Giesbrecht, Kotsireas, and Watt] Corless, R. M., Giesbrecht, M. W., Kotsireas, I. S., Watt, S. M., 2001. Numerical implicitization of parametric hypersurfaces with linear algebra. In: Roanes-Lozano, E. (Ed.), Artificial Intelligence and Symbolic Computation: International Conference AISC 2000. Springer Verlag, Heidelberg, Germany, pp. 174–183.

[Díaz and Kaltofen(1998)] Díaz, A., Kaltofen, E., 1998. FoxBox a system for manipulating symbolic objects in black box representation. In: Gloor, O. (Ed.), Proc. 1998 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'98). ACM Press, New York, N. Y., pp. 30–37.

[Gautschi(1975)] Gautschi, W., 1975. Norm estimates for inverses of Vandermonde matrices. Numerische Mathematik 23, 337–347.

[Gautschi and Inglese(1988)] Gautschi, W., Inglese, G., 1988. Lower bounds for the condition numbers of Vandermonde matrices. Numerische Mathematik 52, 241–250.

[Geddes et al.(1992)Geddes, Czapor, and Labahn] Geddes, K. O., Czapor, S. R., Labahn, G., 1992. Algorithms for Computer Algebra. Kluwer Academic Publ., Boston, Massachusetts, USA.

[Golub et al.(1999)Golub, Milanfar, and Varah] Golub, G. H., Milanfar, P., Varah, J., 1999. A stable numerical method for inverting shape from moments. SIAM J. Sci. Comput. 21 (4), 1222–1243.

[Grigoriev et al.(1990)Grigoriev, Karpinski, and Singer] Grigoriev, D. Y., Karpinski, M., Singer, M. F., 1990. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. SIAM J. Comput. 19 (6), 1059–1063.

[Hildebrand(1956)] Hildebrand, F. B., 1956. Introduction to numerical analysis. McGraw-Hill Book Co., New York.

[Kaltofen and Lakshman Yagati(1988)] Kaltofen, E., Lakshman Yagati, 1988. Improved sparse multivariate polynomial interpolation algorithms. In: Gianni, P. (Ed.), Symbolic Algebraic Comput. Internat. Symp. ISSAC '88 Proc. Vol. 358 of Lect. Notes Comput. Sci. Springer Verlag, Heidelberg, Germany, pp. 467–474.

[Kaltofen and Trager(1990)] Kaltofen, E., Trager, B., 1990. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. J. Symbolic Comput. 9 (3), 301–320.

[Mansour(1995)] Mansour, Y., 1995. Randomized approxmation and interpolation of sparse polynomials. SIAM Journal on Computing 24 (2), 357–368.

[Marple, Jr.(1987)] Marple, Jr., S. L., 1987. Digital Spectral Analysis. Prentice-Hall, Englewood Cliffs, N.J.

[Milanfar et al.(1995)Milanfar, Verghese, Karl, and Wilsky] Milanfar, P., Verghese, G. C., Karl, W. C., Wilsky, A. S., 1995. Reconstructing polygons from moments with connections to array processing. IEEE Trans. Signal Processing 43 (2), 432–443.

[Prony(1795)] Prony, Gaspard-Clair-François-Marie Riche, Baron, 1795. Essai expérimental et analytique sur les lois de la Dilatabilité des fluides élastique et sur celles de la Force expansive de la vapeur de l'eau et de la vapeur de l'alkool, à différentes températures. J. de l'École Polytechnique 1, 24–76.

[Wilkinson(1963)] Wilkinson, J. H., 1963. Rounding errors in algebraic processes. Prentice-Hall, Englewood Cliffs, N.J.

[Zilic and Radecka(1999)] Zilic, Z., Radecka, K., 1999. On feasible multivariate polynomial interpolations over arbitrary fields. In: Dooley, S. (Ed.), ISSAC 99

Proc. 1999 Internat. Symp. Symbolic Algebraic Comput. ACM Press, New York, N. Y., pp. 67–74.

[Zippel(1979)] Zippel, R., 1979. Probabilistic algorithms for sparse polynomials. In: Proc. EUROSAM '79. Vol. 72 of Lect. Notes Comput. Sci. Springer Verlag, Heidelberg, Germany, pp. 216–226.

[Zippel(1990)] Zippel, R., 1990. Interpolating polynomials from their values. J. Symbolic Comput. 9 (3), 375–403.

Schedule of Talks

Thursday 25 March

Time	Programme	
8:30- 9:00 9:00- 9:15	Registration, Coffee Opening	
9:15-10:15	**Invited Lecture**	Dongming Wang *Automated Handling and Proving of Geometric Theorems*
	Coffee	
10:45-11:15 11:20-11:50 11:55-12:25	**Differential Equations**	Ha Le and Ziming Li Sergei Abramov and Denis E. Khmelnov E. Hubert and N. le Roux
	Lunch	
14:00-14:30 14:35-15:05 15:10-15:40	**Representations**	A.N. Prokopenya* Bruce W. Westbury Sergei Haller
	Tea	
16:10-16:40 16:45-17:15 17:20-17:50	**Systems & Packages**	Scott Murray J. Abbott et al Jos Vermaseren
	Dinner	
20:00-21:30	**Evening Presentation**	Bart de Smit *Escher's Print Gallery and the Droste Effect*

Friday 26 March

Time	Programme	
9:15-10:15	**Invited Lecture**	Henk Barendregt *The Quest for Correctness*
	Coffee	
10:45-11:15 11:20-11:50 11:55-12:25	**Geometry**	Dan Roozemond F. Botana and T. Recio Reinier Bröker
	Lunch	
14:00-14:30 14:35-15:05 15:10-15:40	**Symbolic-Numeric**	Thomas Bayer* Hitoshi Yanami and Hirokazu Anai Wen-shin Lee, Mark Giesbrecht and George Labahn
15:45-	Close	