Generalized normal forms and polynomial system solving

Bernard Mourrain, Philippe Trebuchet

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Generalised normal forms and polynomial system solving

B. Mourrain & Ph. Trébuchet

N° 5471

Janvier 2005

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--5471--FR+ENG

Generalised normal forms and polynomial system solving

B. Mourrain & Ph. Trébuchet

Thème SYM — Systèmes symboliques
Projets GALAAD & SALSA

Rapport de recherche n° 5471 — Janvier 2005 — 30 pages

Abstract: This report describes a new method for computing the normal form of a polynomial modulo a zero-dimensional ideal I. We give a detailed description of the algorithm, a proof of its correctness, and finally experimentations on classical benchmark polynomial systems. The method that we propose can be thought as an extension of both the Gröbner basis method and the Macaulay construction. As such it establishes a natural link between these two methods. We have weaken the monomial ordering requirement for Gröbner bases computations, which allows us to construct new type of representations for the associated quotient algebra. This approach yields more freedom in the linear algebra steps involved, which allows us to take into account numerical criteria while performing the symbolic steps. This is a new feature for a symbolic algorithm, which has an important impact on the practical efficiency, as it is illustrated by the experiments at the end of the paper.

Key-words: polynomial equation, normal form, quotient algebra, zero-dimensional ideal, rewrite rule, commutativity, solver, Gröbner basis, resultant

Forme normale généralisée et résolution d'équations polynomiales

Résumé : Ce rapport décrit une nouvelle méthode pour le calcul de forme normale modulo un idéal zéro-dimensionnel I. Nous donnons une description détaillée de l'algorithme, une preuve de sa terminaison, et finalement des expérimentations sur des systèmes testes classiques. Cette méthode peut être vue comme une généralisation du calcul de bases de Gröbner et de la construction de Macaulay. Comme telle, elle établit un lien entre ces deux approches. Nous avons affaibli la condition d'ordre monomiale nécessaire pour les bases de Gröbner, ce qui nous permet de construire de nouveau type de représentations pour l'algèbre quotient associée à l'idéal I. Cette approche apporte plus de liberté dans les étapes d'algèbre line'aire nécessaires au calcul de forme normale, ce qui nous permet de prendre en compte des critères numériques sur les coefficients, tout en travaillant symboliquement sur les polynômes. Cette caractéristique de notre méthode a un impact important sur l'efficacité pratique, comme cela est illustré à la fin du rapport, par nos expérimentations.

Mots-clés : équation polynomiale, forme normale, algèbre quotient, idéal de dimension 0, réécriture, commutativité, résolution, base de Gröbner, résultant

1 Introduction

Solving polynomial systems is the cornerstone in many problems appearing in domains, such as effective (real) algebraic geometry [7, 3], where it is a key ingredient of many algorithms, in robotics and the design of conception tools [6], geometric modeling involving intersection operations, singularity detection [25], signal processing and the design of filters or identification problems [14], chromatology [24], structural molecular biology ...

In this paper, we focus on the following problem: we are given an effective field $\mathbb{K}$, and we consider n-variate polynomials $f_1, \ldots, f_s$ over this field. Our goal is to solve the system of equations $f_1 = 0, \ldots, f_s = 0$ over the algebraic closure $\overline{\mathbb{K}}$ of $\mathbb{K}$.

The ring of n-variate polynomials over $\mathbb{K}$ will be denoted by R, $R = \mathbb{K}[\mathbf{x}] = \mathbb{K}[x_1, \ldots, x_n]$. The polynomials $f_1, \ldots, f_s$ generate an ideal of $\mathbb{K}[\mathbf{x}]$ that we will call I, and in turn I defines the variety $\mathcal{V}$ over $\overline{\mathbb{K}}$, which is the zero set of all the polynomials of I in $\overline{\mathbb{K}}^n$.

Before going further in the description of the solving process, we give some precisions about what we mean by solving a polynomial system. Indeed, if the variety $\mathcal{V}(I)$ contains curves or surfaces then the meaning of *solving* is not clear and we will leave aside this case. In the sequel, we will assume everywhere that *our variety $\mathcal{V}(I)$ is composed of finitely many points $\zeta_1, \ldots, \zeta_d$ (over the algebraic closure of $\mathbb{K}$) of respective multiplicities $\mu_1, \ldots, \mu_d$.*

As the study of algebraic variety started a long time ago, there exists many methods which tackle the problem of solving polynomial systems. They may be classified into two broad families:

- Numerical methods, that proceed by evaluating numerically the equations, that is searching for the 0-level of all these functions.
- Algebraic methods that keep the equations as such and exploit the constraints induced on the unknowns.

In what follows, we will concentrate on this latter family and refer the interested reader to [28] for a study of the other family.

More precisely, algebraic methods for solving a system of polynomial equations $f_1 = 0, \ldots, f_s = 0$ are using special properties of polynomial systems. These properties can all be deduced studying the structure of the quotient $\mathcal{A} = \mathbb{K}[\mathbf{x}]/I$ of the ring of polynomials $\mathbb{K}[\mathbf{x}]$ modulo the ideal $I = (f_1, \ldots, f_s)$. The different algebraic methods produce different types of output:

- An as accurate as needed approximation of the coordinates of the ζ_i.
- A decomposition of the variety into simpler parts (such as triangular systems).
- A Rational Univariate Representation RUR [26] [12] for the roots .

Though these objects are slightly different, we will accept any of them as a valid answer to the problem of describing the solutions of a the polynomial system. A fundamental ingredient we will use to compute one of these objects is the effective description of the quotient algebra $\mathcal{A}$.

Up to now, the algebraic methods have been considered essentially in the case of exact arithmetic over $\mathbb{K}$. In practice, this may induce the treatment of large numbers which size is not related to the complexity of the geometric problems, but to the method used to solve the system. In order to avoid such problems, a new domain of research has emerged at the frontier between symbolic and numeric computation [2], [5], [20], [29], analyzing the numerical quality of the symbolic constraints that are manipulated. A big challenge here is to devise algorithms, which are efficient and numerically stable. We describe here a new method suitable to handle these two requirements. Our concern doing this will be double:

- First, to allow new representations that are stable under the algebraic perturbations leaving the geometric nature of the variety unchanged.
- Next, to allow some speedup in the solving process by computing more general representations avoiding as much as possible artificial computational over-costs.

In this paper, we first recall some classical methods that allow to work in the quotient algebra and show their limitations. Then we show how to combine all those approaches and melt them into a new algorithm, which will be the main result of this paper.

Afterward, we describe how we implemented this new method and treat effectively some examples.

2 From the quotient ring to the roots

For the sake of completeness we recall here some well known results, that show how to recover the information about the roots from the equations. We will limit ourselves to state the theorems without any proof and we refer the interested reader to the references for proofs.

We denote, as mentioned above, by $R = \mathbb{K}[\mathbf{x}] = \mathbb{K}[x_1, \ldots, x_n]$ the ring of n-variate polynomials in the unknowns $x_1, \ldots, x_n$, with coefficients in $\mathbb{K}$, by I is the ideal generated by the polynomials $f_1, \ldots, f_s$, defining the system that we want to solve. The quotient of $\mathbb{K}[\mathbf{x}]$ modulo I will be denoted by $\mathcal{A}$. We will suppose hereafter that I is zero dimensional so that $\mathcal{A}$ is a finite dimensional vector space. The roots, with coordinates in the algebraic closure of $\mathbb{K}$, will be denoted by $\zeta_1, \ldots, \zeta_d$, with $\zeta_i = (\zeta_{i,1}, \ldots, \zeta_{i,n}) \in \overline{\mathbb{K}}^n$.

The dual space of $\mathcal{A}$, that is the set $\mathrm{Hom}_{\mathbb{K}}(\mathcal{A}, \mathbb{K})$ of linear forms from $\mathcal{A}$ to $\mathbb{K}$ will be denoted by $\widehat{\mathcal{A}}$.

Theorem 2.1 *[2], [20], [9] For any element $a \in \mathcal{A}$, consider the operators of multiplication by a:*

$$\begin{array}{rcl} M_a : \mathcal{A} & \to & \mathcal{A} \\ b & \mapsto & a\,b \end{array} \qquad \begin{array}{rcl} M_a^t : \widehat{\mathcal{A}} & \to & \widehat{\mathcal{A}} \\ \Lambda & \mapsto & a \cdot \Lambda = \Lambda \circ M_a. \end{array}$$

Denoting by $\mathbf{1}_\zeta$ the linear form such that for $p \in \mathbb{K}[\mathbf{x}]$, $\mathbf{1}_\zeta(p) = p(\zeta)$, we have the following properties:

- *The eigenvalues of the linear operator M_a (resp. M_a^t) are $\{a(\zeta_1), \ldots, a(\zeta_d)\}$.*
- *The common eigenvectors of $(M_a^t)_{a\in\mathcal{A}}$ are (up to a scalar) $\mathbf{1}_{\zeta_1}, \ldots, \mathbf{1}_{\zeta_d}$.*

This result reduces the numerical solving step to linear algebra computations that are numerically well understood.

When dealing with real algebraic geometry we can also use the following properties.

Theorem 2.2 (Hermite's Quadratic Form) *[1], [27] When $\mathbb{K} = \mathbb{R}$, let $h \in \mathbb{R}[x_1, \ldots, x_n]$, and let Q_h be the quadratic form $Q_h : \mathcal{A} \to \mathcal{A}$, $p \to Tr(hp^2)$. Then we have the following two properties:*

- *The number of complex root ζ_i such as $h(\zeta_i) \neq 0$ is the rank of the quadratic form Q_h.*
- *The number of real root ζ_i such as $h(\zeta_i) > 0$ and the number of real root such as $h(\zeta_j) < 0$ is the signature of Q_h.*

Finally for an exact description of the variety we can use the following result:

Theorem 2.3 *Let $u \in \mathbb{K}[\mathbf{x}]$, $\zeta_1, \ldots, \zeta_k$ be as usual the point composing the variety and $\mu_1, \ldots, \mu_k$ their respective multiplicity. We define the polynomials:*

- $f_u = \Pi_{i=1\ldots k}(T - u(\zeta_i))^{\mu_i}$
- $g_0 = \Sigma_{i=1\ldots k}\mu_i\Pi_{j\neq i}(T - u(\zeta_j))$
- $g_1 = \Sigma_{i=1\ldots k}\mu_i\zeta_1.i\Pi_{j\neq i}(T - u(\zeta_j))$

  $\vdots$

  $g_n = \Sigma_{i=1\ldots k}\mu_i\zeta_k.i\Pi_{j\neq i}(T - u(\zeta_j))$

When u separates the ζ_i, then the preceding family defines a Rational Univariate Representation.

See [13], and [26] for the proofs and for a way to compute these polynomials.

3 Representations of the quotient ring $\mathcal{A}$

In this section, we recall how computations in the quotient ring $\mathcal{A}$ are performed. To be able to compute effectively such a representation of $\mathcal{A}$, we need two things:

- A monomial basis B of $\mathcal{A}$.
- An algorithm to project $\mathbb{K}[\mathbf{x}]$ onto $\langle B\rangle$.

In other words, given a certain $\mathbb{K}$-vector space V with the property that $\mathbb{K}[\mathbf{x}] = V \oplus I$ and a polynomial p, we want to compute the element v of V such that $p - v \in I$.

This implies an isomorphism of $\mathbb{K}$-vector spaces: $\mathcal{A} = \mathbb{K}[\mathbf{x}]/I \sim V$ and gives to V a structure of $\mathbb{K}[\mathbf{x}]$-module. The corresponding normal form N is *the projection of* $\mathbb{K}[\mathbf{x}]$ *onto* V *along* I: $\mathrm{im}(N) = V$, $\ker(N) = I$. A basis B of V will be called a *canonical basis* of $\mathcal{A}$.

The method we propose generalizes Macaulay's constructions and Gröbner bases, taking advantage of the numerical stability of Macaulay's constructions and of the efficiency of Gröbner bases. The following paragraphs are devoted to recall the definitions and algorithms allowing to compute Macaulay's constructions, and Gröbner bases.

- Macaulay construction relies only on linear algebra operations, and also on restrictive mathematical hypotheses on the input polynomial system. As a result this method is very stable under small perturbations, but the too limited range of application of this method lead to find other techniques to represent $\mathcal{A}$.
- In order to circumvent this, Gröbner bases techniques have been developed. Gröbner bases computations can be performed on any polynomial system without any hypotheses on it, and find a representation of $\mathcal{A}$. The improvement of generality of the input system, compared to the preceding method, is obtained at the expense of stability against small perturbations (even if they do not change in any way the geometric nature of the variety $\mathcal{V}(I)$).

Both of these methods produce a so-called normal form algorithm, that is a method to compute a canonical representative in $\mathcal{A}$, as we will see now.

3.1 Resultant based approach

In this subsection we will give a flavor of the ideas behind the resultant based approach with a special emphasis on Macaulay's construction of projective resultants. For more details, see for instance [10], [4]. This construction yields the resultant of $n+1$ homogeneous polynomials over $\mathbb{P}^n$, as the gcd of the maximal minors of a matrix. It can be applied to the computation of multiplication tables, as follows:

Algorithm 3.1 (Macaulay)
INPUT: $f_0 \in \mathbb{K}[\mathbf{x}]$ *whom we want the multiplication operator*
$f_1, \ldots, f_n$ *generic polynomials*
OUTPUT: *The multiplication matrix of* f_0 *in* $\mathcal{A}$

- *Construct sets of monomials* $\mathbf{x}^{E_0}, \ldots, \mathbf{x}^{E_n}, \mathbf{x}^F$ *such that the map*

$$\begin{array}{rcl} S : \langle \mathbf{x}^{E_0} \rangle \times \cdots \langle \mathbf{x}^{E_n} \rangle & \to & \langle \mathbf{x}^F \rangle \\ (q_0, \ldots, q_n) & \mapsto & \sum_{i=0}^n q_i \, f_i \end{array}$$

is surjective and $|E_0| = \prod_{i=1}^n \deg(f_i)$.

- *Compute the matrix of this map and decompose it as:*

$$\begin{array}{c} \begin{array}{cccc} \mathbf{x}^{E_0} f_0 & \mathbf{x}^{E_1} f_1 & \dots & \mathbf{x}^{E_n} f_n \end{array} \\ \left(\begin{array}{c|ccc} \multicolumn{1}{c|}{\mathtt{A}} & \multicolumn{3}{c}{\mathtt{C}} \\ \hline \mathtt{B} & \multicolumn{3}{c}{\mathtt{D}} \end{array} \right) \end{array}$$

- *Check that D is invertible and return the Schur complement*

$$M_{f_0} = A - CD^{-1}B$$

See [18] for more detail.

This method applies for generic input polynomials, for which $\mathbf{x}^{E_0}$ is a basis of $\mathcal{A}$. Similar constructions can also be performed using toric or residual resultants [10], [4]. Such resultant-based methods for solving square polynomial systems rely on well controlled linear algebra steps, and thus are stable against a small deformation of the generic input. But the restriction both in terms of genericity and number of equations of the input system does not allow to treat so many applications. It is then necessary to have a general procedure to compute a monomial basis of $\mathcal{A}$ for any input, and this is done by the well-known method described in the next subsection.

3.2 Gröbner bases

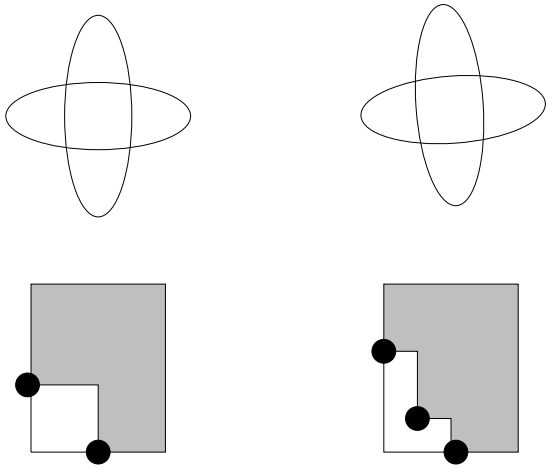
In the previous subsection, we have quickly depicted a method to solve a given system knowing a basis of $\mathcal{A}$. The fact is, that it is **extremely** unusual to know without any prior computation a basis of $\mathcal{A}$! The well known method of Gröbner bases is a way to compute such a basis. It depends on a monomial order γ, which is a total order on the monomials, compatible with the multiplication and such that 1 is smaller than any other monomial. See [7] for more details.

As we can see in [11], [15], [17], instead of doing polynomial algebra, we could do linear algebra, factoring thus many computations. We refer the reader to [11] for an exhaustive explanation about this topic.

Though they algorithmically solve a lot of problems, Gröbner bases have the very serious drawback of needing the *a-priori* introduction of a monomial ordering. The fact that this introduction is done regardless of the geometry of the problem *can*, in some circumstances, lead to artificial unwanted singularities in the representation of $\mathcal{A}$, i.e. structural jumps in the representation unrelated to geometric changes of the solution set. Let us see it on an example of two polynomials

$$\begin{aligned} f_1 &= ax_1^2 + bx_2^2 + n_1(x_1, x_2), \\ f_2 &= cx_1^2 + dx_2^2 + n_2(x_1, x_2), \end{aligned}$$

where $a, b, c, d \in \mathbb{K}$, with $a\,d - b\,c \neq 0$ generic coefficients and $n_1(x_1, x_2), n_2(x_1, x_2) = 0$ are linear terms.



On this picture we have represented the shape of the Gröbner basis computed without any perturbation (on the left) and with a small one $f_1 + \varepsilon_1 x_1 x_2,\ f_2 + \varepsilon_2 x_1 x_2$ (on the right). A structural jump appears in the representation a $\mathcal{A}$. We must also mention that this structural jump comes together with a huge arithmetic over-cost. It would be unavoidable if we were at some singularity where the geometry change but as we can see on the picture and as the theory of resultant shows us, that it is not the case!

The appearing of artificial singularities (due to the algorithm used but not to the geometry of the solutions) can dramatically increase the time and space needed to perform the computation of the basis of $\mathcal{A}$, hindering thus the solving process by artificial arithmetic over-costs.

4 Notations

We recall here some of the definitions stated in [21], [22], [23] and add a few more that we will need in the sequel.

For any subset S of R, we denote by S^+ the set $S^+ = S \cup x_1\, S \cup \cdots \cup x_n\, S$, $\partial S = S^+ \backslash S$. The support $\mathrm{supp}(p)$ of a polynomial $p \in \mathbb{K}[\mathbf{x}]$ is the set of the monomials appearing with non-zero coefficients in p. And for a subset S of another set S', we will denote by S^c the set-theoretical complement of the set S in S'. Given a set S of elements of $\mathbb{K}[\mathbf{x}]$, we will denote by $\langle S \rangle$ the $\mathbb{K}$-vector space spanned by the elements of S. Finally, we will denote to the set of all the monomials in the variables $\mathbf{x} = (x_1, \ldots, x_n)$ by $\mathcal{M}$.

Definition 4.1 *A set B of monomials is said to be connected to* 1 *if and only if for every monomial m in B, there exists a finite sequence of variables $(x_{i_j})_{j\in[1,l]}$ such that*

- $1 \in B$
- $\Pi_{j=1\ldots l'} x_{i_j} \in B,\ \forall l' \in [1,l]$ *and* $\Pi_{j\in[1,l]}\, x_{i_j} \ = m.$

According to this definition, $\{1, x, x^2, x^2y\}$ is connected to 1, but $\{1, x, x^2y\}$ is not connected to 1, since x^2y is isolated from 1.

Definition 4.2 *We say that a set B of monomials is stable by division if for any $m \in B$ and any variable x_i such that $m = x_i\, m'$, we have m' in B.*

Remark that a set B stable by division, is also connected to 1.

Definition 4.3 *Let Λ be a monoid with a good total order $\prec$ on Λ, such that:*

$$\forall \alpha, \beta, \gamma \in \Lambda,\ \alpha \prec \beta \Rightarrow \gamma + \alpha \prec \gamma + \beta$$

A $(\Lambda, \prec)$-graduation of $\mathbb{K}[\mathbf{x}]$ is the decomposition of $\mathbb{K}[\mathbf{x}]$ as the direct sum:

$$\mathbb{K}[\mathbf{x}] = \bigoplus_{\lambda \in \Lambda} \mathbb{K}[\mathbf{x}]_{[\lambda]}$$

with the following property:

$$\forall f \in \mathbb{K}[\mathbf{x}]_{[\alpha]},\ g \in \mathbb{K}[\mathbf{x}]_{[\beta]}, \Rightarrow f.g \in \mathbb{K}[\mathbf{x}]_{[\alpha+\beta]}.$$

We will denote by degree of f or $\deg_\Lambda(f)$, or $\deg(f)$ (when no confusion is possible) or $\Lambda(f)$, the following element of Λ:

$$\Lambda(f) = deg_\Lambda(f) = \min\{\lambda \in \Lambda \mid f \in \bigoplus_{\lambda' \prec \lambda} \mathbb{K}[\mathbf{x}]_{[\lambda']}\}.$$

We also define

- For any set $V \subset \mathbb{K}[\mathbf{x}]$, $V_\lambda = \bigoplus_{\lambda' \prec \lambda}\ \mathbb{K}[\mathbf{x}]_{[\lambda']} \cap V$.
- For any $\lambda \in \Lambda$, $\lambda^+ = \min\{\lambda' \in \Lambda;\ \mathbb{K}[\mathbf{x}]^+_\lambda \subset \mathbb{K}[\mathbf{x}]_{\lambda'}\}$.
- For any $\lambda \in \Lambda$, $\lambda^- = \max\{\lambda' \in \Lambda;\ \mathbb{K}[\mathbf{x}]^+_{\lambda'} \subset \mathbb{K}[\mathbf{x}]_\lambda\}$.

In order not to be confused between different notions of degree, we introduce the following definition:

Definition 4.4 *Let Λ be a graduation of $\mathbb{K}[\mathbf{x}]$, and m a monomial, we define the size of m, denoted by $|m|$, the integer d such that $m = x_{i_1} \cdots x_{i_d}$.*

Obviously, the size of a monomial coincide with its degree when we consider $\mathbb{K}[\mathbf{x}]$ equipped with the standard graduation:

$$\deg_{\mathbb{N}}(x_1^{\alpha_1} \cdots x_n^{\alpha_n}) = \alpha_1 + \cdots + \alpha_n.$$

Then for all $d \in \mathbb{N}$, $\mathbb{K}[\mathbf{x}]_{[d]}$ is the set of homogeneous polynomials of degree d in the variables $x_1, \ldots, x_n$, $d^+ = d + 1$ and if $d > 0$, $d^- = d - 1$ otherwise $d^- = -\infty$.

Another classical graduation is the one associated with a monomial ordering, where $\Lambda = \mathbb{N}^n$ and $\prec$ is a monomial order (see [8][p. 328]) that for all $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, we have

$$\mathbb{K}[\mathbf{x}]_{[\alpha]} = \mathbb{K}\, x_1^{\alpha_1} \cdots x_n^{\alpha_n}.$$

Definition 4.5 *We say that Λ is a reducing graduation if Λ is a graduation and if moreover, we have the property: for all monomials $m, m' \in \mathcal{M}$, such that m' divides strictly m,*

$$\Lambda(m') \prec \Lambda(m),\ \Lambda(m') \neq \Lambda(m).$$

Both graduation induces by the classical degree and a monomial ordering are reducing graduation.

Hereafter, *we will denote by Λ a reducing graduation.*

Definition 4.6 *A rewriting family F for a monomial set B is a set of polynomials $F = \{f_i\}_{i\in\mathcal{I}}$ such that:*

- $\mathrm{supp}(f_i) \subset B^+$,
- *f_i has exactly* **one** *monomial $\gamma(f_i)$ (also called the leading monomial of f_i) in ∂B,*
- *if $\gamma(f_i) = \gamma(f_j)$ then $i = j$,*

Remark that the elements of F can be seen as rewriting rules for the initial monomial using monomial of B.

Definition 4.7 *A reducing family F of degree $\lambda \in \Lambda$ for a set B is a set of polynomials such that:*

- *F is a rewriting family for B,*
- *$\forall m \in \partial B$ of degree at most λ, $\exists f \in F \mid \gamma(f) = m$.*

Example 4.8 *Consider the set $B = \{1, x_0, x_1, x_0x_1\}$, the set of polynomials $F = \{x_0^2 - 1, x_1^2 - x_1, x_0^2x_1 - x_1, x_1^2x_0 - x_1\}$ is a reducing family of degree 3 for this set.*

Notice that a reducing family of degree λ for a set B (connected to 1) allows us to rewrite the monomials of $\langle B^+\rangle_\lambda$, modulo F as elements of $\langle B\rangle_\lambda$. This leads in fact, to the definition of the linear projection R_F, associated to a reducing family for a set B connected to 1.

Definition 4.9 *Given a reducing family of degree $\lambda \in \Lambda$ for a set B connected to 1, we define the linear projection $R_F : \langle B^+\rangle_\lambda \to \langle B\rangle_\lambda$ such that*

$$\begin{array}{l}\forall m \in B_\lambda, R_F(m) = m,\\ \forall m \in \partial B_\lambda, R_F(m) = m - f;\end{array}$$

where $f \in F$ is the unique member of F such as $m = \gamma(f)$. We extend this construction to $\langle B^+\rangle_\lambda$ by $\mathbb{K}$-linearity.

In the sequel, we will make a heavy use of multiplication operators by one variable that we define as follows:

Definition 4.10 *We define*

$$\begin{aligned} M_{i,\lambda} : \langle B\rangle_{\lambda^-} &\to \langle B\rangle_\lambda \\ b &\mapsto R_F(x_i b). \end{aligned}$$

Remark 4.11 *As we can see in the definition the subscript $_\lambda$ is more or less redundant as soon as we know that R_F is constructed from a reducing family of degree λ. This is why, up to a few unavoidable exceptions, we will omit this subscript in the sequel.*

Definition 4.12 *Let $F = \{f_1, \ldots, f_s\}$ be a polynomial set, we denote by $F_{\langle\lambda\rangle}$ the vector space:*

$$F_{\langle\lambda\rangle} = \langle \{x^\alpha f_i |\ \Lambda(x^\alpha f_i) \leq \lambda\}\rangle.$$

Obviously, we have $F_{\langle\lambda\rangle} \subset (F)_\lambda$ where (F) is the ideal generated by F.

Next we introduce a definition, which is weakening the notion of monomial ordering for Gröbner basis:

Definition 4.13 *We say that a function, $\gamma : \mathbb{K}[\mathbf{x}] \to \mathcal{M}$ ($\mathcal{M}$ is the set of all monomials in the unknowns $x_1, \ldots, x_n$), is a choice function refining the graduation Λ, if for any polynomial p, $\gamma(p)$ is a monomial such that*

- $\gamma(p) \in \mathrm{supp}(p)$,
- *if $m \in \mathrm{supp}(p)$, $m \neq \gamma(p)$ then $\gamma(p)$ does not divide m,*
- *and $\Lambda(\gamma(p)) = \max\{\Lambda(m),\ m \in \mathrm{supp}(p)\}$.*

Example 4.14 *In the following, we will consider the so-called* Macaulay *choice function γ refining the degree, such that for all $p \in \mathbb{K}[\mathbf{x}]$, $\gamma(p) = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ satisfies*

- $\deg_{\mathbb{N}}(\gamma(p)) = \max\{\deg_{\mathbb{N}}(m); m \in \mathrm{supp}(p)\} = d$,
- $\exists i_0$ *st.* $\alpha_{i_0} = \max\{\deg_{x_i}(m), m \in \mathrm{supp}(p), \deg_{\mathbb{N}}(m) = d; i = 1, \ldots, n\}$,

Remark 4.15 *The monomial returned by the choice function has the same name as the leading monomial of an element of a reducing family. This is intended, as we will define a reducing family on the behalf of choice functions, and in this framework the two will coincide.*

Hereafter if $S = \{p_1, \ldots, p_s\}$ is a polynomial set, then we denote by

$$\gamma(S) = \{\gamma(p_1) \ldots \gamma(p_s)\}.$$

Definition 4.16 *Let γ be a choice function refining a graduation Λ. For any polynomials $p_1, p_2 \in \mathbb{K}[\mathbf{x}]$, let the C-polynomial relative to γ and (p_1, p_2) be*

$$C(p_1, p_2) = \frac{lcm(\gamma(p_1), \gamma(p_2))}{\gamma(p_1)} p_1 - \frac{lcm(\gamma(p_1), \gamma(p_2))}{\gamma(p_2)} p_2.$$

Let the C-degree of (p_1, p_2) be the degree of $(lcm(\gamma(p_1), \gamma(p_2))/\gamma(p_1))p_1$, and let the leading monomial of the pair (p_1, p_2) be $lcm(\gamma(p_1), \gamma(p_2))$.

Merely, this is the same definition as a S-polynomial [7] when γ is a monomial ordering. We however need a new name to underline that now γ may not be a monomial ordering (i.e. a total order compatible with monomial multiplication). As we will see in the next section, the C-polynomials express commutation conditions for the $M_{i,\lambda}$. Indeed, if up to degree λ the C-polynomials of a given reducing family of degree at most λ vanish, we will see that the $M_{i,\lambda}$ are pairwise commuting.

5 Generalized normal form criterion

Let $F = \{f_1, \ldots, f_s\}$ be a polynomial system and let I be the ideal generated by F. Remember that $F_{\langle\lambda\rangle}$ (the $\mathbb{K}$-vector space spanned by the monomial multiples of the f_i, $x^\alpha f_i$ of degree $\leq \lambda \in \Lambda$) is included in I_λ. Thus, when $I_\lambda = F_{\langle\lambda\rangle}$ we can define a normal form modulo I, up to the degree λ as the projection of $\mathbb{K}[\mathbf{x}]_\lambda$ along $F_{\langle\lambda\rangle}$ onto a supplementary space $\langle B\rangle_\lambda$. Hereafter, we consider a set B of monomials, containing 1.

Let F be a rewriting family, and let $\mathcal{H}$ be the set of their leading monomials, $\mathcal{H} = \{m,\ \exists p \in F, \gamma(p) = m\}$ then, obviously F allows us to define the projection R_F of $B \cup \mathcal{H}$ on B along $\langle F\rangle$. However we may extend this projection using the following extension process:

Definition 5.1 *Let F be a rewriting family, let m be a monomial such that there exists $m' \in \partial B$ and there exists r integers $i_1, \ldots, i_r \in [1, n]^r$, $m = x_{i_1} \cdots x_{i_r} m'$. We define $R^e_F(m)$ by induction on k, as follows.*

- *if $r = 0$, $R^e_F(m')$ is defined as $R^e_F(m') = R_F(m') = m' - f$ where $f \in F$ is such that $\gamma(f) = m'$.*
- *$\forall r \leq k$, $R^e_F(x_{i_1} \cdots x_{i_r} m') = R_F(x_{i_r} R^e_F(x_{i_1} \ldots x_{i_{r-1}} m'))$, if this latter quantity is defined. Otherwise we say that $R^e_F(m)$ is undefined.*

Remark that the above process allows us to define R^e_F only on monomials, and we extend it implicitly, by linearity. Remark also that this extension process is not defined in a unique way. Indeed, two different decompositions of a monomial m may lead to two different values of $R^e_F(m)$. However the following theorem shows that this extension process becomes canonical as soon as we check some commutativity conditions.

Theorem 5.2 *Assume that B is connected to 1. Let F be a rewriting family, and let E be the set of monomials m such that: for all decomposition of m as a product of variables, $m = x_{i_0} \cdots x_{i_k}$, then $R_F(x_{i_0} \cdots R_F(x_{i_k}))$ is defined. Suppose that for all $m \in E$ and all indexes $i, j \in [1, n]$ such $x_i x_j m \in E$, we have:*

$$R^e_F(x_i R^e_F(x_j m)) = R^e_F(x_j R^e_F(x_i m)).$$

Then R^e_F coincides with the linear projection P of $\langle E\rangle$ on $\langle B\rangle$ along the vector space spanned by the polynomials $S = \{x^\alpha f,\ \alpha \in \mathbb{N}^n,\ f \in F$ and $x^\alpha \gamma(f) \in E\}$.

Proof. Remark that the way we define it, makes R_F^e inherently a *linear* multivoque application. Hence to prove the theorem we have to show first that under the above hypotheses, R_F^e becomes a well defined application, and next that this well defined linear application coincide with the projection P of $\langle E\rangle$ on $\langle B\rangle$ along $\langle S\rangle$.

Remark also that E is obviously stable by monomial division: if all the possible decompositions of m as a product of variables are such that $R_F(x_{i_0}\cdots R_F(x_{i_k}))$ is defined, then a fortiori if m' is a divisor of m, this property is true for m'.

Let us show that the extension process defines a univoque application. Let $m = x_{i_0}\, m' = x_{i_1}\, m''$ with $i_0 \neq i_1$ and $m, m', m'' \in E$, then there exists $m''' \in E$ (since E is stable by monomial division) such that $m = x_{i_0}\, x_{i_1}\, m'''$. As $m,\ m',\ m''$, and m''' are in E, $R_F^e(m')$, $R_F^e(m'')$, $R_F^e(x_{i_0}m')$, $R_F^e(x_{i_1}m'')$, and $R_F^e(m''')$ are defined and we have:

$$\begin{aligned} R_F^e(x_{i_0}\, R_F^e(m')) &= R_F^e(x_{i_0}\, R_F^e(x_{i_1} R_F^e(m'''))),\\ R_F^e(x_{i_1}\, R_F^e(m'')) &= R_F^e(x_{i_1}\, R_F^e(x_{i_0} R_F^e(m''))). \end{aligned}$$

The commutation condition guarantees that the two quantities are equal, so that the definition of R_F^e does not depend on the way to write m as a product of variables.

Next we have to show that R_F^e and P coincide on their common set of definition. We do it by induction on the size of the monomials:

It is true that $R_F^e(1) = P(1) = 1$ (since $1 \in B$). For any monomial $m \neq 1$ in E, the property of connectivity of B and the definition of E gives us: $\exists m' \in E$ and $i_0 \in [1, n]$ such that $m = x_{i_0} m'$ and $R_F^e(m')$ is defined, so that we have:

$$R_F^e(m) = R_F^e(x_{i_0} m') =^{def} R_F^e(x_{i_0} R_F^e(m')) =^{induction} R_F^e(x_{i_0} P(m')) \in \langle B\rangle.$$

Now by induction, $m' - P(m') \in S_{\lambda^-}$ and

$$m - R_F^e(m) = x_{i_0}(m' - P(m')) + (x_{i_0} P(m') - R_F^e(x_{i_0} P(m'))) \in \langle S\rangle.$$

Thus $R_F^e(m)$ is the projection of m on $\langle B\rangle$ along $\langle S\rangle$.

□

Suppose now that we are given a reducing family of degree λ instead of a rewriting family. Then we can further extend the above theorem with the help of the following lemma.

Lemma 5.3 *Let F be a reducing family of degree λ for a set B connected to 1, and suppose that $\forall f \in F,\ \Lambda(\gamma(f)) = \Lambda(f)$. With the notation of theorem 5.2, the set E of monomials m such that for all decomposition of m as a product of variables, $m = x_{i_0}\cdots x_{i_k}$, then $R_F^e(x_{i_0}\cdots R_F^e(x_{i_k}))$ is defined, contains the set of monomials of degree less or equal to λ.*

Proof. Let $m \in \mathcal{M}_\lambda$ be a monomial of degree less or equal to λ, then m can be written as $m = x_{i_1}\ldots x_{i_d}$ with $d = |m|$.

Let us prove by induction on $k \leq d$, that $p_k = R_F(x_{i_k} R_F(\cdots R_F(x_{i_1}))\cdots)$ is defined and that $\deg_\Lambda(p_k) \leq \deg(x_{i_k}\cdots x_{i_1})$.

As F is a reducing family of degree λ, for $m' \in \mathrm{supp}(x_{i_{k+1}} p_k) \cap \partial B$, we have a rewriting rule for m'. The hypothesis that $\forall f \in F,\ \Lambda(\gamma(f)) = \Lambda(f)$ implies that m' rewrites in terms of monomials of degree bounded by $\deg_\Lambda(x_{i_{k+1}} \cdots x_{i_1})$. This proves that $p_{k+1} = R_F(x_{i_{k+1}} p_k)$ is defined and that $\deg_\Lambda(p_{k+1}) \leq \deg_\Lambda(x_{i_{k+1}} \cdots x_{i_1})$

Finally, we deduce that $R^e_F(x_{i_1} \cdots R^e_F(x_{i_d}))$ is defined, for any decomposition $m = x_{i_1} \cdots x_{i_d} \in \mathcal{M}_\lambda$ so that $m \in E$. This ends the proof. □

Theorem 5.4 *Let F be a reducing family of degree λ for a set B connected to 1. If we have:*

- $\forall f \in F,\ \Lambda(\gamma(f)) = \Lambda(f)$.
- $M_{j,\lambda} \circ M_{i,\lambda^-} = M_{i,\lambda} \circ M_{j,\lambda^-}$, *for* $1 \leq i, j \leq n$,

then, we can extend R_F to a linear projection R^e_F from $\mathbb{K}[\mathbf{x}]_\lambda$ onto $\langle B\rangle_\lambda$ of kernel $F_{\langle\lambda\rangle}$.

Proof. As F is a reducing family of degree λ, by lemma 5.3, we have $E \supset \mathcal{M}_\lambda$.

Let us prove that for all $m \in \mathcal{M}_{\lambda^{--}}$ and all pair of indexes (i, j), there exists a way to define R^e_F such that:

$$R^e_F(x_i\, R^e_F(x_j m)) = R^e_F(x_j\, R^e_F(x_i m)).$$

As $\mathcal{M}_{\lambda^{--}} \subset \mathcal{M}_\lambda \subset E$, $R^e_F(m)$ is defined so that $\mathrm{supp}(R^e_F(m)) \subset B$.

We define $R^e_F(x_i m) = R_F(x_i R^e_F(m))$ and $R^e_F(x_j m) = R_F(x_j R^e_F(m))$.

With this definition we have:

$$R^e_F(x_i R^e_F(x_j m)) = M_{i,\lambda}(M_{j,\lambda^-}(R^e_F(m))) = M_{j,\lambda}(M_{i,\lambda^-}(R^e_F(m))) = R^e_F(x_j R^e_F(x_i m)).$$

which proves the commutation property. We end the proof by applying theorem 5.2. □

We directly deduce from the preceding result, the following property:

Corrolary 5.5 *With the hypothesis of theorem 5.4, we have $\mathbb{K}[\mathbf{x}]_\lambda = \langle B\rangle_\lambda \oplus F_{\langle\lambda\rangle}$.*

Let us give here an even more effective way to check that we have a projection from $F_{\langle\lambda\rangle}$ (vector space spanned by the monomial multiples of the f_i of degree λ) onto $\langle B\rangle_\lambda$ (element of degree λ of the vector space spanned by B) starting from a reducing family of degree λ, without computing *explicitly* the multiplication operators.

Theorem 5.6 *Let $\lambda \in \Lambda$. Let F be a reducing family of degree λ, for B. Assume that $\forall\ f \in F,\ \Lambda(\gamma(f)) = \Lambda(f)$ and let R_F be the induced reduction from $\langle B^+\rangle_\lambda$ onto $\langle B\rangle_\lambda$. Then $\forall f, f' \in F_{\langle\lambda\rangle}$ such that $C(f, f') \in \langle B^+\rangle_\lambda$,*

$$R_F(C(f, f')) = 0$$

iff R_F extends uniquely as a projection R^e_F from $\mathbb{K}[\mathbf{x}]_\lambda$ onto $\langle B\rangle_\lambda$ such that $\ker(R^e_F) = F_{\langle\lambda\rangle}$.

Proof. By theorem 5.4, we have to show that this condition is equivalent to the commutation of the operators $M_{i,\lambda'}, \lambda' < \lambda$ on the monomials of $B_{\lambda--}$.

For any $m \in B_{\lambda--}$ and any $i_1 \neq i_2$ such that $x_{i_1} m \in \partial B$, $x_{i_2} m \in \partial B$, there exists $f, f' \in F_{\langle \lambda^- \rangle}$ such that $\gamma(f) = x_{i_1} m$, $\gamma(f') = x_{i_2} m$. Thus, we have $R_F(x_{i_1} m) = \gamma(f) - f$, $R_F(x_{i_2} m) = \gamma(f') - f'$ and $C(f, f') = x_{i_2} f - x_{i_1} f' \in \langle B \rangle^+_\lambda$. Consequently,

$$\begin{aligned} & M_{i_2,\lambda}(M_{i_1,\lambda^-}(m)) - M_{i_1,\lambda}(M_{i_2,\lambda^-}(m)) \\ = \;& M_{i_1,\lambda}(\gamma(f) - f) - M_{i_2,\lambda}(\gamma(f') - f) \\ = \;& R_F(x_{i_1}\gamma(f) - x_{i_1} f) - R_F(x_{i_2}\gamma(f') - x_{i_2} f') \\ = \;& R_F(x_{i_2} f' - x_{i_1} f) = R_F(C(f', f)). \end{aligned}$$

which is zero by hypothesis. A similar proof applies if $x_{i_1} m \in B$ or $x_{i_2} m \in B$.

Conversely, since $\ker(R^e_F) = F_{\langle \lambda \rangle}$ and $C(f, f') \in F_{\langle \lambda \rangle} \cap \langle B \rangle^+_\lambda$, we have $R_F(C(f', f)) = R^e_F(C(f', f)) = 0$, which proves the equivalence and theorem 5.6. □

Remark 5.7 *In the proof, we have shown that if the C-polynomials up to the degree λ reduce to 0, then the multiplication operators $M_{i,\lambda}$ commute.*

Finally, this leads to a new proof of theorem 3.1 of [21]:

Theorem 5.8 *Let F be a reducing family of all degrees $\lambda \in \Lambda$ for a set B of monomials, connected to 1, let R_F be the corresponding reduction from $\langle B^+ \rangle$ onto $\langle B \rangle$, and let $M_i : \langle B \rangle \to \langle B \rangle$ such that $\forall\, b \in \langle B \rangle$, $M_i(b) = R_F(x_i\, b)$. Then,*

$$M_j \circ M_i = M_i \circ M_j, \text{ for } 1 \leq i, j \leq n$$

iff there exist a unique projection R which extends uniquely to a linear projection R^e_F from $\mathbb{K}[\mathbf{x}]$ onto $\langle B \rangle$ such that $\ker(R^e_F) = (F)$ and $(R^e_F)_{|\langle B^+ \rangle} = R_F$.

Proof. Under these hypotheses, by theorem 5.4, for any $\lambda \in \Lambda$, $(R_F)_{|\langle B^+ \rangle_\lambda}$ extends uniquely to a projection $R^e_{F_\lambda}$ from $\mathbb{K}[\mathbf{x}]_\lambda$ onto $\langle B \rangle_\lambda$, such that $\ker(R^e_{F_\lambda}) = F_{\langle \lambda \rangle}$. Since for any $\lambda, \lambda' \in \Lambda$ such that $\lambda < \lambda'$, we have $(B_{\lambda'})_\lambda = B_\lambda$, and $F_{\langle \lambda \rangle} \subset (F_{\langle \lambda' \rangle})_\lambda$. We also have $F_{\langle \lambda \rangle} = (F_{\langle \lambda' \rangle})_\lambda$ so that $(R^e_{F_{\lambda'}})_{|\mathbb{K}[\mathbf{x}]_\lambda} = R^e_{F_\lambda}$. This defines a unique linear operator R^e_F on $\mathbb{K}[\mathbf{x}]$ such that $R^e_{|\mathbb{K}[\mathbf{x}]_\lambda} = R^e_{F_\lambda}$ and $\ker(R^e_F) = \sum_{\lambda \in \Lambda} F_{\langle \lambda \rangle} = (F)$. It proves the direct implication. The converse implication is immediate. □

6 The algorithm

The algorithm, that we describe now, consists in computing a suitable set B for a basis of the quotient ring $\mathcal{A} = \mathbb{K}[\mathbf{x}]/I$ and reducing rules, in order to project onto $\langle B \rangle$ along I. It can be interpreted as the *check* that the constructed B is a basis of $\mathcal{A}$. The method proceeds incrementally, until a fixed point is reached.

Here are some notations, used in the description of the algorithm:

- k will be the loop index at which we will consider the families of polynomials.
- P_k will be the set of polynomials, from which a reducing family will be constructed.
- $M_k = \gamma(P_k)$ will be the set of monomials, used as the leading terms of the family P_k.
- $(P_k|M_k)$ will be the matrix of coefficients of the monomials of M_k in the polynomials of P_k, written row by row.
- *X:=PseudoSolve(A,B)* is the multiplication by **a** pseudo inverse (if it exists) so that $A\,X = B$.
- The function $SelectMinDeg_\gamma$ applied to a polynomial set S is a function that implements to following algorithm:
  - Set $d = min(\Lambda(g), g \in S)$
  - Construct the list $S' = \{f \in S, \Lambda(f) = d\}$
  - for $f \in S'$ do
    - apply γ on f to get a monomial m.
    - Perform linear combinations between the elements of S' to eliminate m of the support of the other elements of S'.
    - if those linear combinations produce polynomials of degree less than d return $selectMinDeg_\gamma$ applied to S'
  - if no decreasing of the degree of the members of S' occurs, return S' together with the monomial set $\gamma(S')$ constructed during the for loop.
- By *Increment* B, we mean adding to B the monomials corresponding to the zero columns in the matrix $(P_k|M_k)$.
- By monomials at Hamming distance 1 of B, we mean the monomials that we obtain from the monomials of B by writing them as a product of variables and changing *exactly* one index in this writing.
- By neighbors of the monomials added to B, we mean the polynomials whose leading term is at Hamming distance 1 of the set added to B by *Increment* .

Notice that solving a system of the form $(P_k|M_k)X = P_k$ is equivalent (when $(P_k|M_k)$ is invertible) to computing a vector of polynomials, with one "leading monomial" in M_k and all the other outside.

We also need the following subroutines:

Algorithm 6.1 REMAINDER. *Let p be a polynomial, and F a family of valid rewriting rules with respect to B. Let $r := rem(p, F, B)$ be a polynomial such that* $\mathrm{supp}(r) \cap \gamma(F) = \emptyset$ *and* $r \in \langle p, F\rangle$.

Algorithm 6.2 REDUCTION. *Reduce(P, F, B) reduces the set of polynomials P by F a family of valid rewriting rules with respect to B. It works like this:*
For all $p \in P$, for all $m \in \mathrm{supp}(p)$, if $m \notin B$ do the following:

1) $r_0 := 1$;

2) decompose m into a product of variables $m = x_{i_1} \dots x_{i_d}$;

3) for j from 1 to $d = |m|$ do

a) $r_j = x_{i_j} * r_{j-1}$,
b) $r_j = rem(r_j, F, B)$,

end for

4) substitute m by r_d in p;

Remark 6.3 *It is very important to notice here that the definition of Reduce is* not *canonical but depends on some particular choice of a writing of the monomials as products of variables.*

Let us describe now the main algorithm:

Algorithm 6.4 NORMAL FORM.
INPUT: *$F = f_1, \dots, f_s$ generating an ideal I of dimension 0, and γ a choice function refining a reducing graduation Λ.*
INITIALIZATION:
$P_0 = SelectMinDeg_\gamma(f_1, \dots, f_s)$, *poolpol* $= \{f_1 \dots f_s\} \backslash P_0$, $B = (\gamma(f_i))^c$,
$k = 1$.
CORE LOOP: DO

1) $Cont_k = \{p$, *polynomials in poolpol s.t.* $\mathrm{supp}(p) \subset B\}$.

2) For $p \in Cont_k$, remove from B the monomial ideal generated by $\gamma(p)$.

3) $P_k = \{$ *the valid rewriting rules in poolpol* $\} \backslash P_{k-1}$.

4) Compute $C_{k+1} = \{C(f, f')$ *such that* $\gamma(C(f, f')) = x_{i_0}\gamma(f) = x_{i_1}\gamma(f')$, $f, f' \in P_k\}$, *poolpol* $=$ *poolpol* $\cup\, C_{k+1}$.

5) Compute $P_{k+1} = \partial P_k \cap B^+$.

6) $M_{k+1} = \partial\gamma(P_k) \cap B^+$.

7) $P_{k+1} :=$ *PseudoSolve* $((P_{k+1}|M_{k+1}), P_{k+1})$.

8) $C_{k+1} :=$ *Reduce*$(C_{k+1}, \cup_{i \le k+1} P_j, B)$ *and set poolpol* $=$ *poolpol* $\cup\, C_{k+1}$.

9) $r = \#M_{k+1} - Rank((P_{k+1}|M_{k+1}))$, *if* $r \neq 0$ *increment* B *and add to* P_{k+1} *the neighbor of the monomials* B *is incremented with.*

10) $poolpol = poolpol \cup P_{k+1}$.

11) For all $i \in [1, k+1]$, $P_i = \{$*the reduced polynomials in* $P_i\}$.

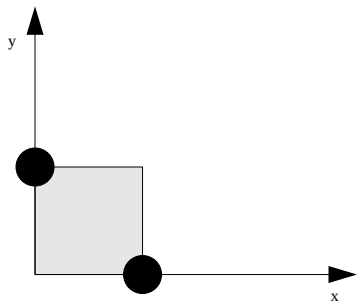
12) $poolpol = Reduce(poolpol, \bigcup_{i=1..k+1} P_i, B)$; $k = k+1$.

WHILE $poolpol \neq \emptyset$;
OUTPUT: $\{P_j, j = 0 \dots k\}$ *a reducing family for all* $k \in \mathbb{N}$

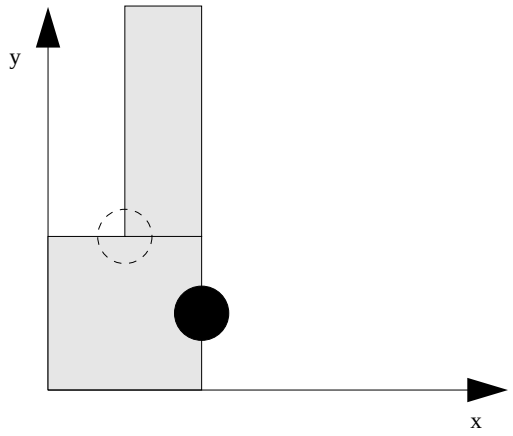
Let us actually see how this algorithm works on an example:

Example 6.5 *Let the input system be* $F = \{x^2 + yx, y^2 + xy, x^4 - 1\}$, *and let* γ *be a Macaulay-like choice function, i.e. choosing monomial with highest partial degree. The reducing graduation refined being here just the classical one.*

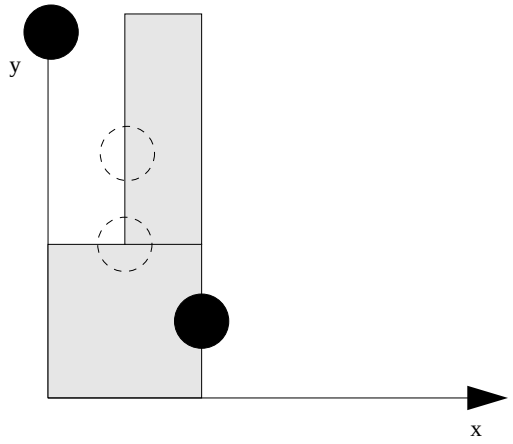
After the initialization $k = 0$, $P_0 = \{x^2 + xy, y^2 + xy\}$, $M_0 = \{x^2, y^2\}$ *and* B *is just* $\{1, x, y, xy\}$. *We have omitted from* P_0 *the polynomial* $x^4 - 1$ *because it is not reduced. Graphically the situation is:*



Next we perform the $^+$ *operation and obtain the matrix* $(P_1|M_1) = \begin{pmatrix} 1, 1 \\ 1, 1 \end{pmatrix}$, *matrix which is rank deficient, we have* $r \neq 0$. *Consequently we increment* B*: we add to* B *the monomial* xy^2, *and we add the neighbors, namely* $xy^2 + y^3$ *to* P_1.



We undertake here the second turn is the `Core` *loop. As* x^4-1 *is still non-reduced, we do not add it to* P_1*. Next we perform another* $^+$ *operation, on* P_1*. The same situation as previously occurs (indeed we are just considering the preceding system times* y*). We increment* B *by* xy^3 *and add to* P_2 *the polynomial* xy^3+y^4*.*



We undertake here the third turn is the `Core` *loop. This time* x^4-1 *is reduced and is a member of* $Cont_2$*, we apply* γ *to it and exclude from* B *the monomial ideal* xy^3*.* P_2 *is then composed of* 3 *polynomials:* $x^2y^2+xy^3, y^4+xy^3, xy^3-1$*. Their is then two polynomials in*

C_3: $x^2y^3 - x$ and $xy^4 - y$. However P_3 is empty, since no multiple of the polynomials of P_2 by a variable stay in B^+. Applying Reduce to poolpol makes $x - y$ appear.

The next steps are only a repetition of what we explained earlier and finally lead to the basis: y^4, x. Our reducing family is thus: $\{\mathbf{y^4} - 1, \mathbf{x} - y\}$, and with it we can compute normal forms for any polynomials of $\mathbb{K}[\mathbf{x}]$.

Definition 6.6 *For $k \in \mathbb{N}$, we will denote by B^k the set B computed at loop k of the algorithm 6.4, and by $F^k = \cup_{i=0}^{k} P_i$.*

Definition 6.7 *In this section, we will denote by R^k the operator such that, given $p \in \mathbb{K}[\mathbf{x}]$, $R^k(p)$ returns $Reduce(p, F^k, B^k)$.*

Definition 6.8 *We say that a polynomial $f \in \mathbb{K}[\mathbf{x}]$ is reducible at loop k, if $R^k(f) \in \langle B^k \rangle$. The polynomial f is reduced at loop k, if $f \in \langle B^k \rangle$.*

Let us also mention now some useful facts:

Remark 6.9

1. *During the algorithm 6.4, at loop k, $\forall f \in F^k$, $\gamma(f) \in \partial B^k$, since the polynomials in P_i are reduced (step 12).*
2. *In the algorithm 6.4 (step 9 and definition of the function increment), B is constructed as*
$$B = \cup_i \cup_{f \in F_i} \{(m_i) \backslash (\gamma(f))\}, \tag{1}$$
where $m_i \in M$.
3. *At step 9, the monomials added to B, are by construction, in M_{k+1}.*
4. *For all monomial $m \notin B$, there exists a monomial m' also in ∂B such that $m'|m$ and m' is the leading monomial of a polynomial of F^k. This comes from the fact that B is of the form (1).*

We will now prove that the previous algorithm stops and produces a correct result, i.e. a reducing family for all degrees. To do so, we need the following lemmas and proposition.

Lemma 6.10 *If m is a monomial, and $r = R^k(m)$ for some index $k \in \mathbb{N}$, then $\forall m' \in \mathrm{supp}(r), \Lambda(m') \leq \Lambda(m)$.*

Proof. This comes from the definition of R^k (see algorithm 6.2): given a monomial m, it computes a list of variables whose product equals m, it adds to a variable r, which initially equals to 1, linear combination of polynomials belonging to F^k, and multiply the resulting polynomial by one variable of the list, and so on until the list is exhausted. Since Λ is a graduation, we have with the notation of algorithm 6.2, $\Lambda(r_j) \leq \Lambda(x_{i_1} \cdots x_{i_j})$ for all loop j of the reduction algorithm, so that $\forall m' \in \mathrm{supp}(r_j), \Lambda(m') \leq \Lambda(m)$. □

Lemma 6.11 *For all $d \in \mathbb{N}$, there exist finitely many P_k containing polynomials whose leading monomials are of size less than d.*

Proof. Remark that if a set P_k contains a polynomial whose leading monomial is of size 0, then the algorithm stops as we know that this ideal I is in fact $\mathbb{K}[\mathbf{x}]$: any non constant monomial is strictly greater than 1 for any choice function, so having 1 as leading monomial means that the polynomial itself is 1.

Next, remark that P_k contains a polynomial whose leading monomial is of size d, if and only if we have excluded from B a monomial of size d.

Finally remark that the monomials added to B at loop k, are in $M_{k+1} \subset \partial M_k$.

The lemma is true for $d = 0$. Assume that it is true for $d-1$, and let k_0 be the last index such that P_{k_0} contains a polynomial whose leading monomial is of size $d-1$.

Thus, after loop k_0 no monomial of size d can be added to B. Hence, since the P_k are auto-reduced there will be at most as many $P_k,\ k \geq k_0$ containing polynomial with leading monomials of size d, as there are monomials of size d, due to the second remark.

This proves the result by induction. □

Definition 6.12 *Given a size d, we will call l_d the lowest integer such that after loop l_d no $P_k,\ k \geq d$ contain polynomials whose leading monomials is of size less than d.*

Notice here that at step 12, some polynomials $f \in$ *poolpol* may not be reducible (see definition 6.8), depending on the shape of B and $\mathrm{supp}(f)$. However, we have the following property:

Proposition 6.13 *Given $f \in \mathbb{K}[\mathbf{x}]$ there exists k such that f is reducible by applying R^k and such that $\forall m \in \mathrm{supp}(R^k(f)),\ k \geq l_{|m|}$.*

Proof. We split $\mathrm{supp}(f)$ into l sets $\Lambda_0,\ldots,\Lambda_l$ such that

$$\Lambda_i = \{m \in \mathrm{supp}(f) \setminus \cup_{j=0}^{i-1} \Lambda_j, \Lambda(m) = \min(\Lambda(m'), m' \in \mathrm{supp}(f) \setminus \cup_{j=0}^{i-1} \Lambda_j)\}.$$

We will show that there exists a certain loop k at which all the monomials of all the sets Λ_i are simultaneously reducible.

We consider here the set $S_0 = \Lambda_0$, and define $k_0 = l_{|f|}$. For all $m \in \Lambda_0$, consider the computation of $R^{k_0}(m) = \mathit{Reduce}(m, F^{k_0}, B^{k_0})$ described in definition 6.2. We denote by $r_1, \ldots, r_d$ the sequence of remainders which are computed at loops (b).

- Either, for all $i = 1 \ldots d$, we have

  $$\mathrm{supp}(r_i) \subset B^{k_1},$$

  where $k_1 = \max(l_{|r_j|\, j=1..d}, k_0)$. We have that $r_d \in B^{k_1}$ and that $k_1 \geq l_{|r_d|}$, therefore implying that m is reducible at loop k_1 and that for all $k \geq k_1$, m is also reducible at loop k with the property that $R^k(m) = r_d$.

- Or, there exists $m \in \Lambda_0$ and $i_0 < d$ such that $\mathrm{supp}(r_i) \subset B^{\max_{j\le i}(l_{|r_j|},k_0)}$, but not $\mathrm{supp}(r_{i_0+1})$. Let $k_1 = \max(l_{|r_j|}\, j = 1..i_0, k_0)$.

  Then for $m' \in \mathrm{supp}(r_{i_0})\backslash B^{k_1}$, we have, by lemma 6.10:

  $$\Lambda(m') \le \Lambda(x_{i_1}\cdots x_{i_0}) < \Lambda(x_{i_1}\cdots x_{i_d}) = \Lambda(m),$$

  since Λ is a reducing graduation (see definition 4.5). This implies that $m' \notin (m)$.

  Now let $S_1 = S_0 \cup \{m' \in \mathrm{supp}(r_{i_0})\backslash B^{k_1},\ \Lambda(m')\ minimal\}$. The above construction of m' implies that $S_0 \subset S_1$ and, that $S_0 \neq S_1$.

  We iterate this construction, replacing S_0 with S_1, S_2 with S_1, ... Since $\mathbb{K}[\mathbf{x}]$ is Noetherian, the sequence of S_i is finite.

Consequently, we have proved that in both cases, there exists an index k_0' such that all the monomials of Λ_0 are reducible after loop k_0'.

Let $red(\Lambda_0)$ be the set of all these reductions, so that k_0' satisfies the property $k_0' \ge l_{max(|r|,\ r\in red(\Lambda_0))}$.

Consider the next sets Λ_i. By the same process, there exists an index k_i such that all the monomials of Λ_i are reducible after loop k_i. Let $red(\Lambda_i)$ be the set of all these reductions, so that k_1 also satisfies the property $k_i \ge l_{max(|r|,\ r\in red(\Lambda_i))}$.

Hence, for $k \ge \max(k_i)$ we have that all the monomials of all the sets Λ_i are simultaneously reducible, i.e. f is reducible at loop k with the property that $\forall m \in \mathrm{supp}(R^k(f)),\ k \ge l_{|m|}$. □

Corrolary 6.14 *Let $d \in \mathbb{N}$, there exists an index k_d such that:*

- *For all m of size at most d, $\mathrm{supp}(R^{k_d}(m)) \subset B^{k_d}$.*
- *and for all m of size at most d, all $m' \in \mathrm{supp}(R^{k_d}(m))$ of size d', $k \ge l_{d'}$.*

Proof. This is a trivial application of the above proposition. Let m be a monomial of size at most d, and let k_m be the index exhibited above. Then defining $k_d = max(k_m,\ |m| \le d)$, we obtain the result. □

Remember now that for any $k \in \mathbb{N}$, the definition of $R^k(m)$ relies on some particular choices for the writing of m as a product of variables. As there is only finitely many monomials of size less than d and as for a given monomial, there exists only finitely many ways to write it as a product of variables. Thus, we can define the following quantity:

Definition 6.15 *For any $d \in \mathbb{N}$, we define k_d^* as the maximum of the index k_d for all the possible definitions of $R^k(m)$.*

Lemma 6.16 *Let m be a monomial, and f_i be one of the generators of I then there exists a loop index k of the algorithm such that mf_i reduces to 0.*

Proof. From the above lemma there exists a loop k of the algorithm such that mf_i is reducible at loop k. Let us prove now that when mf_i is reducible it reduces to 0. It is true that f_i reduces to 0. Suppose that for all m' of size $d-1$ $m'f_i$ reduces 0 at a loop k_1, and let x_j be a variable. By definition of $R_{k_1}(x_j m' f_i) = R_{k_1}(x_j R_{k_1}(m' f_i)) = R_{k_1}(x_j 0) = 0$. Hence this induction on the size of m shows that if mf_i is reducible then it reduces to 0. This ends the proof. □

Lemma 6.17 *For any $d \in \mathbb{N}$, at loop l_d, F^{l_d} is reducing family with respect to B^{l_d}, for the set of monomials of size $\leq d$.*

Proof. We need to prove that $\forall m \in \partial B^{l_d}$ of size $\leq d$, there exists one polynomial $p \in F^{l_d}$ such that $m = \gamma(p)$.

By remark 6.9, as $m \in \partial B^{l_d}$, there exists $p_1 \in F^{l_d}$ such that $m = x_{i_1} \cdots x_{i_l} \gamma(p_1)$ and $\forall l' \leq l,\ x_{i_1} \cdots x_{i_{l'}} \gamma(p_1) \in \partial B^{l_d}$. Let k be the smallest integer for which there exists such a $p_1 \in F^k$. By lemma 6.16, $x_{i_1} p_1$ will be reducible at loop $k_1 \geq k$, but as $x_{i_1} p_1$ is computed from p_1 at step 5, there exists a polynomial $p_2 \in F^k$ such that $\gamma(p_2) = x_{i_1} \gamma(p_1)$.

With the same arguments, for $j \leq l$, there exist $k_j \in \mathbb{N}$ and a polynomial $p_j \in F^{k_j}$ such that $\gamma(p_j) = x_{i_1} \cdots x_{i_j} \gamma(p)$. Hence there exists a polynomial $p \in F^{k_l}$ such that $\gamma(p) = m$. However as $\gamma(p)$ is of size d, we have by definition of l_d, $k_l \leq l_d$ and $p \in F^{l_d}$.

This proves that F^{l_d} is a reducing family with respect to B^{l_d}, for the monomials of size $\leq d$. □

Proposition 6.18 *Assume that the ideal I generated by the elements $f_1, \ldots, f_m$ is zero-dimensional. Then the algorithm 6.4 stops and yields a basis B of $\mathcal{A} = R/I$ and a normal form R onto $\langle B \rangle$, modulo I.*

Proof. Let us first prove that B is connected to 1 at each loop of the algorithm. To check this, remark that if B is connected to 1, it remains so when we *increment* it (step 9). Indeed a monomial added to B is in B^+. Moreover if B is connected to 1, and if we remove from B some monomial ideals in the step 2, B remains so. Finally, as B is connected to 1 at the beginning of the algorithm, it remains so all along the computation, accordingly to the above statement.

We show now that the algorithm stops. To do this we prove that at some loop k, we have computed a representation of the quotient algebra for the monomials of size d.

Let d_1 be the maximal size of a monomial appearing in $R^{k_d^*}(m)$ for all m of size at most d.

Remark that step 8 and lemma 6.12 guaranty that all the C-polynomials whose C-leading term is of size less than d_1 will simultaneously reduce to 0 for all the possible definitions of R^k at $k \leq k_{d_1}^*$ (see definition 6.15).

Hence, for any definition of R^{k_1}, any monomial m of size less than $d-2$, and any variables x_i, x_j $R^{k_1}(x_j R^{k_1}(x_i m)) = R^{k_1}(x_i R^{k_1}(x_j m))$. Hence, by theorem 5.2, we deduce the following points:

- we can extend the projection of ∂B^{k_1} (restricted to monomials of size less than d) on B^{k_1} along $\langle F^{k_1}\rangle$ to the projection of the vector space spanned by the monomials of size less than d on B^{k_1} along $\langle F^{k_1}\rangle$.
- this extension coincide with R^{k_1} on their common set of definition.

Now as I is zero dimensional, R/I is a $\mathbb{K}$-vector space of finite dimension d_0. Suppose the algorithm does not stop before the loop l_{d_0+1}. Then in $B^{l_{d_0+1}}$ there is at least d_0+1 monomials of size less than d_0+1 since $B^{l_{d_0+1}}$ is connected to 1. Hence as $\dim(R/I)=d_0$ there is a linear dependence relation between these monomials in R/I. In other words there is a nonzero polynomial $p=\Sigma_{i=1..s}q_if_i\in I$ whose support is in $B^{l_{d_0+1}}$ and does not involve monomials of size greater than d_0+1. Let d be the greatest size of the monomials appearing in $\mathrm{supp}(q_if_i)$. The above paragraph shows that there exists an index k such that R^k coincide with the projection from the vector space spanned by the monomials of size less or equal to d to B^k along $\langle F^k\rangle$. Hence as all of the q_if_i reduce to 0 by R^k and as R^k coincide with the projection on the vector space spanned by the monomials of size less than d, p will eventually reduce to 0. This means that there will be a polynomial of one of the P_l, $l\geq l_{d_0+1}$ whose leading monomial is one of the monomials of p, this contradicts the fact that $\mathrm{supp}(p)\subset B_{d_0+1}^{l_{d_0+1}}$. Therefore, the algorithm cannot go beyond loop l_{d_0+1}.

Remark now that at the loop k_0 where the algorithm stops F^{k_0} is a reducing family of all degree $\lambda\in\Lambda$ for the set B^{k_0} satisfying the property that $\forall p\in F^{k_0},\ \gamma(p)=max\{\gamma(m),\ m\in\mathrm{supp}(p)\}$ as γ refines the reducing graduation Λ.

By step 8 (the C-polynomials reduce to 0) remark 5.7, and by theorem 5.4, we can extend R^{k_0} to the projection $R^*_{F_\lambda}$ for all $\lambda\in\Lambda$ and thus to the projection R^* of $\mathbb{K}[\mathbf{x}]$ onto B along I such that $R^*_{|B^+}=R^{F^*}$. □

We must notice that there is far too many polynomials computed at each loop, more precisely:

Remark 6.19 *If, up to loop k, the C-polynomials reduce to zero, then to compute the polynomials in P_k from those in P_{k-1}, we only need to compute one polynomial per monomial introduced in M_k (for each monomial m in M_k, one polynomial whose leading monomial is m) since two polynomials of P_k with the same leading term could be combined to form a C-polynomial.*

Thus the algorithm can be modified to take into account this remark: the construction of P_{k+1} will involve only polynomials with distinct leading terms.

We can also notice that only a small fraction of the computed C-polynomials are actually needed, namely those whose leading monomial is minimal for the division.

We can now examine what happen in the important case where γ is a monomial ordering:

Proposition 6.20 *Assume that γ is a monomial order, then at every loop k of modified algorithm, the matrix $(P_{k+1}|M_{k+1})$ is of full rank.*

Proof. Due to the modification (remark 6.19) we are in the case where only one polynomial of P_{k+1} per monomial of M_{k+1} is computed.

We order the monomials of M_{k+1} with respect to the total order γ, and the polynomials in P_{k+1} according to their leading terms.

Since for all $p \in P_{k+1}$ and $m \in \text{supp}(p)$, $\gamma(p) > m$, and as the polynomials of P_{k+1} have distinct leading terms, the matrix $(P_{k+1}|M_{k+1})$ is triangular. □

7 Experimentations

Unless otherwise stated, the computations are performed on an athlon 2400+ with $256Mo$ of main memory. We expose here the results obtained with our implementation of the particular case where the graduation we use for $\mathbb{K}[\mathbf{x}]$ is the usual one. In the sequel, *dlex* will refer to the choice function associated to the Degree lexicographical order, *dinvlex* to the degree inverse lexicographical order, *random* to the choice function that returns randomly any of the monomials of maximum degree of the polynomial given as its input, *mac* to the choice function that returns the monomial of maximal degree with highest partial degree (Macaulay's choice function), *minsz* to the choice function over the rational that minimize the memory needed in the reduction loop, and *mix* to the choice function that return either the result of *minsz* applied to its input, or the result of *dlex* applied to its input.

7.1 Generic equations

We examine here the behavior of the method on the *Katsura equations*[1]. These equations are projective complete intersection with no zero at infinity, i.e. we can apply the Macaulay revisited techniques of [22]. We can compare the timings between the former implementation [22] and the present program, the comparison being pertinent since the objects computed are the same. To do so, we use `long double` for the arithmetic type. As we know a priori what monomials will be leading monomials for the whole computation we can guarantee that no test to 0 return erroneous result.

number of variables	Athlon 2400+		UltraSparc 10	
4	0.0s	3M	0.02	2M
5	0.02s	3M	0.08s	2M
6	0.09s	3M	0.24s	2.3M
7	0.42s	4.5M	0.95s	3.6M

Those timings compare favorably with those given in [22]. Now we compare the timings when we take other choice functions (the computation is now performed using modular arithmetic).

[1] `http://www-sop.inria.fr/galaad/data/`

n	mac	random	dlex
6	0.14s	0.18s	0.13s
7	0.69s	1.21s	0.76s
10	95.16s	3219.85s	300.15s
11	662.23s	∞	2162.78s

This table shows how bad it can be choosing the wrong monomial, and how interesting it is to know which one to choose. Numerically we observe that choosing the *mac* function also results in a better conditioning of the computations. More precisely on Katsura(6) we have.

choice function	number of bits	time	$max(\|\|f_i\|\|_\infty)$
dlex	128	1.48s	10^{-28}
dinvlex	128	4.35s	10^{-24}
mac	128	1s	10^{-30}
dlex	80	1.35s	10^{-20}
dinvlex	80	3.98s	10^{-15}
mac	80	0.95s	10^{-19}
dlex	64		–
dinvlex	64		–
mac	64	0.9s	10^{-11}

For the 64 bits computation the results computed for the *dlex* and *dinvlex* orders are erroneous due to roundoff errors.

The time given is the time spent in the computation of the multiplication matrices. Afterward, we used either LAPACK to perform the eigenvector computations or Maple when we needed extended precision. Because of the different nature of these tools, we do not report on the solving part timing.

Finally we show here the amount of memory needed to perform the computations over $\mathbb{Q}$, using GMP `mpq`.

	mac	minsz	dlex	mix
time	4.50s	10.21s	7.39s	9.83s
size	4.2M	4.1M	4.4M	4.1M

7.2 Parallel robot

Let us consider the famous direct kinematic problem of the parallel robot [16], [19]. We will consider as a test system, the system given on the web page `http://www-sop.inria.fr/galaad/data/`. First we use floating point numbers to check to numerical requirements of the computations for different orders. For testing a number to be 0, we will use a leveling (here 10^{-8} is enough) and we will check afterward that the choices performed are the same as those done using modular arithmetic. This is equivalent to the use an hybrid arithmetic.

choice function	number of bits	time	$max(\|\|f_i\|\|_\infty)$
dlex	128	9.13s	$0.3 * 10^{-24}$
dinvlex	128	11.1s	$0.3 * 10^{-23}$
mac	128	9.80s	$0.1 * 10^{-24}$
dlex	250	11.16s	$0.42 * 10^{-63}$
dinvlex	250	13.8s	$0.135 * 10^{-60}$
mac	250	11.62s	$0.46 * 10^{-63}$

Here we see that choosing the right choice function can increase (but not so much in this case) the numerical accuracy of the roots. We mention here that when using 80 bits of precision, only Macaulay's order gives a correct result, but not the other choice functions.

number of bits	time	$max(\|\|f_i\|\|_\infty)$
80	9.80s	10^{-6}
128	9.80s	10^{-24}
250	11.62s	10^{-63}
500	19s	10^{-140}

Finally, we performed tests using rational coefficients.

	mac	minsz	dlex	mix
time	327s	378.73s	367.65s	508.50
size	20M	20M	30M	25M

In fact, it is not so surprising to see that the choice function γ has an enormous impact in terms of the computational time and of the memory required. We also mention here that over constraining the system can result in dramatic decrease of the computation time. Indeed expressing more constraints on the rotation than necessary gives additional quadratic equations that simplifies lot the computations here are the results we obtain with such a redundant parametrization:

arithmetic	time	memory
250 bits	$1.5s$	8M
128 bits	$1.2s$	6M

7.3 Cyclic equations

The two previous sections seems to show that in all the cases it is advantageous to consider a Macaulay like strategy. We should moderate this assertion as the $Cyclic(n)$[2] family shows.

n	dinvlex time	dlex time	mac time
5	0.12s	0.17s	0.23s
6	1.45s	0.92s	6.27s
7	365.99s	170.3s	1356.11s

Needless to say that the size of the output also vary a lot with those choices functions.

[2] http://www-sop.inria.fr/galaad/data/

7.4 Chromatography

We have also tested our implementation on centrifugal partition chromatography problems [24]. We refer to [24] or [30] for a detailed exposition of the equation arising and give our results:

Arithmetic	Time	Memory	$max(\|f_i(\zeta_j\|))$
double	$0.01s$	1M	error
128 bit	$0.04s$	1M	10^{-12}
$\mathbb{Q}$	$0.13s$	2M	10^{-12}

The conclusion of those experiences is that the numerical accuracy of the results is more limited by the numerical step following the normal forms computation than by the normal forms computations themselves.

8 Conclusion and future work

The method described above opens new possibilities for computing normal forms, and consequently, for solving polynomial systems. Its force resides in the fact that the choices in the normal form algorithm are less constrained than in the previously known methods. But some work remains:

- the algorithm is guaranteed to stop only when the ideal I is zero dimensional. How can we modify the stop test in the *Core While* to detect positive dimensional ideals and how can we build N in that case?
- The fact that we have to reduce *poolpol* in the algorithm should be computationally expensive.
- An efficient implementation of the general method is developed as an evolution of the one described in [23]. It is available in the SYNAPS library[3]

References

[1] M.E. Alonso, E. Becker, M.F. Roy, and T. Wörmann. Zeros, multiplicities and idempotents for zero dimensional systems. In L. González-Vega and T. Recio, editors, *Algorithms in Algebraic Geometry and Applications*, volume 143 of *Prog. in Math.*, pages 1–15. Birkhäuser, Basel, 1996.

[2] W. Auzinger and H. J. Stetter. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. In *Proc. Intern. Conf. on Numerical Math.*, volume 86 of *Int. Series of Numerical Math*, pages 12–30. Birkhäuser Verlag, 1988.

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

[3] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real ALgebrac Geometry*. Springer, 2003.

[4] L. Busé, M. Elkadi, and B. Mourrain. Using projection operators in computer aided geometric design. In *Topics in Algebraic Geometry and Geometric Modeling,*, pages 321–342. Contemporary Mathematics, 2003.

[5] R.M. Corless, P.M. Gianni, and B.M. Trager. A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots. In W.W. Küchlin, editor, *Porc. ISSAC*, pages 133–140, 1997.

[6] S. Corvez and F. Rouillier. Using computer algebra tools to classify serial manipulators. In F. Winkler, editor, *Automated Deduction in Geometry*, volume 2930 of *Lecture Notes in Artificial Intelligence*, pages 31–43. Springer, 2003.

[7] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer Verlag, New York, 1992.

[8] D. Eisenbud. *Commutative Algebra with a view toward Algebraic Geometry*, volume 150 of *Graduate Texts in Math.* Berlin, Springer-Verlag, 1994.

[9] M. Elkadi and B. Mourrain. *Introduction à la résolution des systèmes d'équations algébriques*, 2003. Notes de cours, Univ. de Nice (310 p.).

[10] I.Z. Emiris and B. Mourrain. Matrices in Elimination Theory. *J. of Symbolic Computation*, 28(1&2):3–44, 1999.

[11] J.-C. Faugère, 1994. Personal Communication.

[12] Marc Giusti, Grégoire Lecerf, and Bruno Salvy. A Gröbner free alternative for polynomial system solving. *Journal of Complexity*, 17(1):154–211, 2001.

[13] L. Gonzalez-Vega, F. Rouillier, and M.F. Roy. *Symbolic Recipes for Polynomial System Solving*. Some Tapas of Computer Algebra. Springer, 1997.

[14] O. Grellier, P. Comon, B. Mourrain, and Ph. Trébuchet. Analytical blind channel identification. *IEEE Trans. on Signal Processing*, 50(9):2196–2207, 2002.

[15] D. Lazard. Grœbner basis, Gaussian elimination and resolution of algebraic equations. *Lec. Notes in Comp. Sci.*, 162, 1983.

[16] D. Lazard. Stewart platforms and gröbner bases. In *ARK'92*, Proceedings of Advance in Robot Kinematik, Ferrare, Italia, September 1992.

[17] H. Lombardi. Un nouvel algorithme de calcul de base de gröbner, 1998. Rapport tech. Université Franche Comté.

[18] F.S. Macaulay. Some formulae in elimination. *Proc. London Math. Soc.*, 1(33):3–27, 1902.

[19] B. Mourrain. The 40 generic positions of a parallel robot. In M. Bronstein, editor, *Proc. Intern. Symp. on Symbolic and Algebraic Computation*, ACM press, pages 173–182, Kiev (Ukraine), July 1993.

[20] B. Mourrain. Computing isolated polynomial roots by matrix methods. *J. of Symbolic Computation, Special Issue on Symbolic-Numeric Algebra for Polynomials*, 26(6):715–738, Dec. 1998.

[21] B. Mourrain. A new criterion for normal form algorithms. In M. Fossorier, H. Imai, Shu Lin, and A. Poli, editors, *Proc. AAECC*, volume 1719 of *LNCS*, pages 430–443. Springer, Berlin, 1999.

[22] B. Mourrain and Ph. Trébuchet. Solving projective complete intersection faster. In C. Traverso, editor, *Proc. Intern. Symp. on Symbolic and Algebraic Computation*, pages 231–238. New-York, ACM Press., 2000.

[23] B. Mourrain and Ph. Trébuchet. Algebraic methods for numerical solving. In *Proc. of the 3rd International Workshop on Symbolic and Numeric Algorithms for Scientific Computing'01 (Timisoara, Romania)*, pages 42–57, 2002.

[24] J. M. Nuzillard, J. H. Renault, M. Maciuk, M. Zeches-Hanrot, R. Margraff, and P .Trébuchet. Benzalkonium chloride as a strong anion exchanger in centrifugal partition chromatography. In *Pitcom*, 2002.

[25] M. P. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer Verlag, 2002.

[26] F. Rouillier. *Algorithmes efficaces pour l'étude des zéros réels des systèmes polynomiaux*. PhD thesis, Université de Rennes, 1996.

[27] M.F. Roy. Basic algorithms in real algebraic geometry: from Sturm theorem to the existential theory of reals. In *Lectures on Real Geometry in memoriam of Mario Raimondo*, volume 23 of *Exposition in Mathematics*, pages 1–67, 1996.

[28] Bl Sendov, A. Andreev, and N. Kjusrkiev. *Handbook of Numerical Analysis*, volume III. Elsvier, 1994. Solution of Equations in $\mathbb{R}^n$ (part 2).

[29] Hans J. Stetter. *Numerical polynomial algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2004.

[30] Ph. Trébuchet. *Vers une résolution stable et rapide des équations algébriques*. PhD thesis, Université Pierre et Marie Curie, 2002.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399