Algebraic methods for numerical solving

Presented at 4th Int. Workshop ...

B. Mourrain* & Ph. Trébuchet*

Abstract. In this paper, we present a new approach for computing normal forms in the quotient algebra \( A \) of a polynomial ring \( R \) by an ideal \( J \). It is based on a criterion, which gives a necessary and sufficient condition for a projection onto a set of monomials, to be a normal form modulo the ideal \( J \). This criterion does not require any monomial ordering and generalises the Buchberger criterion of \( S \)-polynomials involved in Gröbner basis computation. It also leads to new algorithms for constructing the multiplicative structure of a zero-dimensional algebra based on intrinsic operations on vector spaces of polynomials. We also report on a symbolic-numeric software environment that we have developed to test these ideas and on experimentation of this new approach on practical problems, in computer vision and signal processing.

Key words: Algebraic Solving, Gröbner basis, normal form, eigenvalue, numerical solving.

AMS Subject Classification:

1. Introduction

Polynomial systems appear naturally in many applicative domains going from Computer Aided Design to robotics, including signal processing, computer vision etc. Hence the solving process appears as major issue in this broad range of applications.

An approach which is now classical consists in introducing a total order on the set of monomials, compatible with the multiplication, and in computing the so-called Gröbner basis of the ideal \( J \) generated by \( f_1, \ldots, f_m \) for this monomial ordering. In this approach, the polynomials are seen as rewriting rules on the monomials and the associated Gröbner basis gives a way to compute the normal form modulo \( J \), of any element in \( \mathbb{K}[x_1, \ldots, x_n] \). A basis of \( A \) is given by the set of monomials outside the initial of the ideal and the multiplicative structure is obtained through normal form computations (Cox et al. (1992)), (Becker et al. (1993)).

Once this normal form is known, one can solve the polynomial system, either by using symbolic methods like elimination of variables (Cox et al. (1992)), (Faugère et al. (1993)), (Emiris and Mourrain (1999)), rational representation (Macaulay (1912))[p. 88], (Renegar (1992)), (Alonso et al. (1996)), (Giusti et al. (1995)), (Rouillier (1996))...and solving a univariate polynomial equation or by matrix manipulations and eigenvalue or eigenvector computations (Lazard (1981)), (Auzinger and Stetter (1988)), (Möller (1993)), (Mourrain (1998)), ...

In practice, many of the polynomial systems are given with approximate coefficients, which means that we have to consider not just a single polynomial system but its neighbourhood. Despite this, the aforementioned methods proceed pointwise and do not take into account the continuity that exists in this neighbourhood.

Our motivation here is to develop methods which can exploit as much as possible, the continuity in the coefficients of the input system. It is well-known that the geometry of the solutions is not a global continuous function of the parameters of a polynomial system. However, in many practical situations such as in robotics, in computer vision, ..., the geometry of the solutions is not changing in a neighbourhood of the input system. We are interested in algorithms, which can take into account this stability. From a practical point of view, such a requirement is necessary for developing safe and numerically stable methods for solving polynomial equations with approximate coefficients. We propose to achieve this goal by widening the range of possible bases that we can use to compute normal forms.

* Partially supported by ECG IST-2000-26473 and GAIA IST-2001-35512
In this paper, we present a new method for computing the normal form with respect to a set of polynomials, provided it generates a zero dimensionnal ideal. The construction is based on the criterion stated in (Mourrain (1999)), which gives a necessary and sufficient condition for a projection onto a monomial set, to be a normal form modulo the ideal \( I \) generated by the equations that we want to solve. A precise description of the algorithm is given and its termination and correctness are proved. Finally, we illustrate this approach on two practical applications, one in computer vision for the autocalibration of a camera based on Kruppa's equations and another one in signal processing for blind identification of sources.

2. Algebraic methods for solving polynomial equations

We consider the situation, where \( I = \langle f_1, \ldots, f_m \rangle \subset R = \mathbb{K}[x_1, \ldots, x_n] \) and \( A = R/I \) is of finite dimension \( D \) over \( \mathbb{C} \) (= \( \mathbb{K} \) the algebraic closure of \( \mathbb{K} \)). This implies that the set of solutions \( Z(I) = \{ \zeta \in \mathbb{C}^n; p_1(\zeta) = \cdots = p_m(\zeta) = 0 \} \) is finite: \( Z(I) = \{ \zeta_1, \ldots, \zeta_d \} \) with \( d < D \) (the variety \( Z(I) \) is 0-dimensional). Equality in \( A \) is denoted by \( \equiv \).

Let \( \widehat{R} \) (resp. \( \widehat{A} \)) denotes the space of linear forms on \( R \) (resp. \( A \)). The special elements of \( \widehat{R} \) which evaluate polynomials at points \( \zeta \in \mathbb{C}^n \) are denoted by \( 1_\zeta : p \mapsto p(\zeta) \). Hereafter, we will identify \( \widehat{A} \) with the set of elements \( \Lambda \in \widehat{R} \) such that \( \Lambda(I) = 0 \) (also denoted by \( I^\perp \)). Thus, \( 1_\zeta \in \widehat{A} \) iff \( \zeta \in Z(I) \).

Given an element \( a \in A \), we consider the multiplication \( M_a \) by \( a \) in \( A \) and its transposed \( M_a^t \):

\[
M_a : A \to A \quad M_a^t : \widehat{A} \to \widehat{A}
\]

\[
b \mapsto ab \quad \Lambda \mapsto a \cdot \Lambda = \Lambda \circ M_a.
\]

Matrix methods for solving polynomial equations are based on the following proposition (Auzinger and Stetter (1988)), (Möller (1993)), (Mourrain (1998)):

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

This proposition reduces the resolution problem to an eigenvalue problem. Such a computation can be performed with approximate coefficients (the error analysis being well-understood (Golub and Van Loan (1996))) and is implemented in easily available libraries such as LAPACK (Anderson et al. (1990)).

Another way to solve the system consists in exploiting the properties of the so-called Chow form

\[
\Delta(u) = \det(u_0 + u_1 x_1 + \cdots + u_n x_n) = \prod_{\zeta \in Z(I)} (u_0 + u_1 \zeta_1 + \cdots + u_n \zeta_n)^{\mu(\zeta)},
\]

where \( \mu(\zeta) \) is the multiplicity of the root \( \zeta \). This leads for instance to rational univariate representation techniques, which expresses the roots of the input system as the image of the root of a univariate polynomials by a rational map (Macauley (1912))[p. 88], (Renegar (1992)), (Alonso et al. (1996)), (Giusti et al. (1995)), (Rouillier (1996)), (Rouillier (1999)). The resolution of the input system reduces to solving a univariate polynomial and substituting into the rational functions. This method is also used to compute the isolated points of a variety, or more generally its geometric decomposition (Giusti et al. (2001)), (Elkadi and Mourrain (1999)).

Many other approaches for solving polynomial equations exists such as triangular set techniques (Aubry et al. (1999)), homotopy (Verschelde et al. (1994), Li (1997)), optimisation, ... In this paper, we concentrate ourself on algebraic methods which yield a complete description of the quotient structure \( A \).

2.1. Resultant based methods. Resultant based methods exploit the properties of matrices used to construct the resultant over the projective space \( \mathbb{P}^n \) (Macauley (1902)), over a toric varieties (Gelfand et al. (1994)Gelfand, Kapranov, and Zelevinsky), (Canny and Emiris (1993)), or more generally over residual intersections (Busé et al. (2000)), (Busé et al. (2001)Busé, Elkadi, and Mourrain). Such a resultant matrix
$S$ usually comes from a Sylvester map of the form (Emiris and Mourrain (1999)):

$$S : \langle x^{E_0} \rangle \times \cdots \times \langle x^{E_n} \rangle \rightarrow \langle x^F \rangle$$

$$(q_0, \ldots, q_n) \mapsto \sum_{i=0}^n q_i f_i,$$

where $f_0, \ldots, f_n \in R := \mathbb{K}[x]$ (or $L := \mathbb{K}[x^\pm]$ in the toric case), $x^{E_i}$, $i = 1, \ldots, n$ and $x^F$ are sets of monomials in the variables $x$ (or $x^\pm$ in the toric case). These sets of monomials are chosen in such a way that $x^F$ contains the subset $x^{E_0}$, and $x^{E_0}$ is a basis of $A = R/(f_1, \ldots, f_n)$ (resp. $A = L/(f_1, \ldots, f_n)$ in the toric case), for generic values of the coefficients of the polynomials $f_i$. Therefore, the matrix $S$ can be divided into 4 blocks:

$$S = \begin{bmatrix} E_0 & E_1 & \cdots & E_n \\ \hline A & B \\ C & D \end{bmatrix},$$

where the columns $[A^\tau, C]^\tau$ represent monomial multiples of $f_0$, those of $[B^\tau, D]^\tau$ represent monomial multiples of $f_1, \ldots, f_n$.

**Proposition 2.** (Mourrain and Pan (2000)) When the polynomials $f_0, \ldots, f_n$ are generic (that is when $D$ is invertible), the set $x^{E_0}$ is a basis of $A = R/(f_1, \ldots, f_n)$ and the matrix of multiplication by $f_0$ in this basis of $A$ is

$$M_{f_0} = A - B D^{-1} C.$$

According to this proposition, the system $f_1 = \cdots = f_n = 0$ can be solved in the generic case. It is then enough to compute the resultant matrix $S$, compute the Schur complement of $D$, and finally compute the eigenvalues (or eigenvectors) of this Schur complement either explicitly (Emiris and Mourrain (1999)), or implicitly (Bonyfalat et al. (1998)). The structure of these matrices (their sparsity or their quasi-Toeplitz structure) can be exploited in such algorithms (Mourrain and Pan (2000)).

Another interest of this approach is that the formula for $M_{f_0}$ is continuous in the coefficients of $f_0, f_1, \ldots, f_n$. Such a construction can thus be applied with approximate coefficients, for we can control the error in terms of the condition number of $D$.

In Macaulay’s construction (Macaulay (1902)), which applies for generic dense polynomials, the subsets $x^{E_i}$ are constructed as follows. Let $d_i = \deg(f_i)$ and $\nu = \sum_{i=0}^n d_i - n$. Let $x^F$ be the set of all monomials in $x$ of degree lower than $\nu$. It contains $N = (\nu+n \choose n)$ monomials. Let $x^d_0 x^{E_0}$ be the set of all monomials of $x^F$ which are divisible by $x^d_0$. Among the remaining monomials in $x^F - x^d_0 x^{E_0}$, let $x^d_0 x^{E_0}$ be the set of monomials which are divisible by $x^{d_0}$. Similarly, for $i = n-2, \ldots, 1$, we define by induction $x^d_i x^{E_i}$ to be the set of monomials of $x^F - x^{d_0} x^{E_0} - \cdots - x^{d_i+1} x^{E_{i+1}}$ which are divisible by $x^{d_i}$. The set $x^F - x^d_n x^{E_0} - \cdots - x^d_1 x^{E_1}$ is denoted by $x^{E_0}$ and is equal to

$$B := x^{E_0} = \{ x^{d_1} \cdots x^{d_n} : 0 \leq \alpha_i \leq d_i - 1 \}.$$ 

It has $d_1 \cdots d_n$ monomials.

The size of the linear system (of the form $DX = V$) that we have to solve in order to compute the matrix $M_{f_0}$, is therefore $N - \prod_{i=1}^n d_i = (\nu+n \choose n) - \prod_{i=1}^n d_i$. Using Stirling formula, we can check that this size is $O(e^n d^n)$ where $d = \max_i (d_i)$.

Here is a table giving the number of solutions, the size of the Macaulay matrix and the size of the matrix $D$ to invert, for a system of generic polynomial equations of degree 2 in $n$ variables:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>D</td>
<td>S</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>462</td>
<td>430</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>1 716</td>
<td>1 652</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>6 435</td>
<td>6 307</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>24 310</td>
<td>24 054</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>92 378</td>
<td>91 866</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
<td>352 716</td>
<td>351 692</td>
</tr>
<tr>
<td>11</td>
<td>2048</td>
<td>1 352 078</td>
<td>1 350 030</td>
</tr>
</tbody>
</table>
These figures show the limitation of such an approach: even if the matrices involved in this computation are structured or sparse (Mourrain and Pan (2000)), (Bondyfalat et al. (1998)), it will not be so easy to handle linear systems of size 10^6.

2.2. Gröbner basis methods. One of the customary tools used nowadays to compute normal forms consists in introducing a total order on the set of monomials and in computing the so-called Gröbner basis of the ideal generated by \( f_1, \ldots, f_m \) for this monomial ordering. This monomial ordering is compatible with the multiplication, so that the polynomials can be seen as rewriting rules on the monomials. The associated Gröbner basis (deduced by completion of the initial rewriting rules) gives a way to compute the normal form of any element in \( \mathbb{K}[x_1, \ldots, x_n] \) modulo the ideal \( I \). These techniques were introduced by B. Buchberger (Buchberger (1965)) (see also (Hironaka (1964))). He was a student of W. Gröbner (Gröbner (1970)), who asked him how to compute the multiplication tables in a quotient algebra.

From the Gröbner basis computation, we deduce many interesting informations on the quotient algebra. Indeed, a basis of \( \mathcal{A} \) is given by the set of monomials outside the initial of the ideal and the multiplicative structure is obtained by reduction by the computed rewriting rules (Cox et al. (1992)), (Becker et al. (1993)).

However, this approach is not very adapted to polynomials with approximate coefficients, as we will see below. Consider for instance a system of equations in two variables of the form

\[
\begin{align*}
p_1 &:= a x_1^2 + b x_2^2 + l_1(x_1,x_2) = 0 \\
p_2 &:= c x_1^2 + d x_2^2 + l_2(x_1,x_2) = 0
\end{align*}
\]

where \( a, b, c, d \in \mathbb{C} \) are complex numbers, \( a d - b c \neq 0 \) and \( l_1, l_2 \) are linear forms. Form a Gröbner basis of these polynomials for a monomial order refining the degree order. The initial ideal is generated by \( (x_1^2, x_2^2) \) and the corresponding basis of \( \mathcal{A} = \mathbb{C}[x_1, x_2]/(p_1, p_2) \) is \( \{1, x_1, x_2, x_1 x_2\} \).

![Fig. 1. The two conics with horizontal and vertical axis and the basis \((1, x_1, x_2, x_1 x_2)\) of \( \mathcal{A} = \mathbb{C}[x_1, x_2]/(p_1, p_2) \), deduced from a Gröbner basis computation for a monomial ordering refining the degree ordering.](image)

Consider now a small perturbation of this system

\[
\begin{align*}
\tilde{p}_1 &= p_1 + \epsilon_1 x_1 x_2 \\
\tilde{p}_2 &= p_2 + \epsilon_2 x_1 x_2,
\end{align*}
\]

where \( \epsilon_1, \epsilon_2 \in \mathbb{C} \) are “small” parameters. The zero-set is also the points of intersection of two conics, which are slightly deformed but the initial is \( (x_1^2, x_1 x_2, x_2^2) \) (if \( x_1 \succ x_2 \)).

In the result of a small perturbation, basis may “jump” from one set of monomials to another, though the two situations are very closed to each other from a geometric point of view. Moreover, some of the polynomials of the Gröbner basis have large coefficients, for we have to divide by coefficients of the order \( \epsilon_1, \epsilon_2 \). Thus, this computation has to be carried out with exact and multi-precision arithmetic, inducing an unnecessary additional cost in the resolution process.

In fact, computing a Gröbner basis for a given monomial order only allows “flat” deformations of the initial ideal of the input system (see (Eisenbud (1994))). In this work, we want to consider more general deformations, by removing the constraints induced by the monomial ordering.

3. Generalized normal form algorithms
3.1. Notations. We recall some of the definitions stated in (Mourrain (1999)), (Mourrain and Trébuchet (2000)), and add a few more that we will need in the sequel.

We denote by $R = \mathbb{K}[x] = \mathbb{K}[x_1, \ldots, x_n]$ the ring of multivariate polynomials in the variables $x_1, \ldots, x_n$, with coefficients in $\mathbb{K}$. For any subset $S$ of $R$, let $S^+ = S \cup x_1 S \cup \cdots \cup x_n S$. For any $k \in \mathbb{N}$, $S(k)$ is the subset of $S$ of elements of degree less than or equal to $k$, and $S[k]$ will denote the set of elements of $S$ of degree exactly $k$. Finally $\langle S \rangle$ will denote the vector space spanned by $S$. The support $\text{Supp}(p)$ of a polynomial $p \in \mathbb{K}[x]$ is the set of monomials with non-zero coefficients in $p$. And for a subset $S$ of another set $S'$, we will denote by $S^c$ the complement of the set $S$ in the set $S'$.

Hereafter, by normal form we mean the following: the computation of a representative in a vector space $V \subset R$, for any class $a$ of the quotient algebra $A = \mathbb{K}[x]/I$. This implies an isomorphism of vector spaces: $A = \mathbb{K}[x]/I \sim V$ and gives to $V$ a structure of $\mathbb{K}[x]$-module. The corresponding normal form $N$ is the projection of $\mathbb{K}[x]$ onto $V$ along $I$ (im$(N) = V$, ker$(N) = I$). A basis $B$ of $V$ will be called a canonical basis of $A$.

**Definition 1.** A set $B$ of monomials is said to be connected to 1 if and only if 1 $\in B$, and if $m \in B$ of degree strictly non-negative, there exists $x_j$ such that $x_j$ divides $m$ and $m/x_j \in B$.

**Definition 2.** A normalising family of degree $k$ for a monomial set $B$ connected to 1 is a set of polynomials $(f_i)_{i \in I}$ such that:

- $\text{Supp}(f_i) \subset B^+$,
- $f_i$ has exactly one monomial (named $\gamma(f_i)$) in $B^+ \backslash B$,
- if $\gamma(f_i) = \gamma(f_j)$ then $i = j$,
- $\forall m \in B^+ \backslash B$ of degree at most $k$, $\exists i \in I$ such that $\gamma(f_i) = m$.

A normalising family of degree $k$ for a monomial set $B$ allows to construct a projection from $\langle B^+ \rangle(k)$ onto $B$ as follows:

**Definition 3.** Given a normalising family of degree $k$ for a set $B$ connected to 1, we can define the linear projection $N_{(k)} : \langle B^+ \rangle(k) \to \langle B \rangle$ such that

- $\forall m \in B(k), N_{(k)}(m) = m$,
- $\forall m \in B^+(k) \backslash B(k), N_{(k)}(m) = m - f$ where $m = \gamma(f)$ and $f \in F$.

We extend this construction to $\langle B^+ \rangle(k)$ by linearity.

**Definition 4.** We define $M_{i,k} : B_{(k-1)} \to B$

$\alpha^* \mapsto N_{(k)}(x_i \alpha^*)$

**Definition 5.** Let $F = \{f_1, \ldots, f_n\}$ be a normalising family of degree $k$ for a (finite) monomial set $B$, then we denote by $F_{(k)}$ the vector space: $F_{(k)} = \Sigma_{i=1}^n (f_i)_{(k)}$.

We have $F_{(k)} \subset I(k)$ where $I$ is the ideal generated by $F$. When these two sets coincide, we can define a normal form modulo $I$, up to the degree $k$.

**Theorem 1.** Let $F$ be a normalising family of degree $k$ for a monomial set $B$ connected to 1. If

- $\text{deg}(f) = \text{deg}(\gamma(f)), \forall f \in F$.
- $M_{j,k} \circ M_{i,k-1} = M_{i,k} \circ M_{j,k-1}, for 0 < i,j < n + 1$,
then we can extend $N_{(k)}$ uniquely to a linear projection $N$ from $\mathbb{K}[x]_{(k)}$ onto $\langle B \rangle_{(k)}$ such that $N(F_{(k)}) = 0$.

Proof. As the operator we want to construct is a linear projection, it is enough to define it on the monomials and then to extend it by linearity.

First of all, remark that the compositions $M_{j,k} \circ M_{i,k-1}$ and $M_{i,k} \circ M_{j,k-1}$ are well defined. Indeed as $\forall f \in F$, $deg(\gamma(f)) = deg(f)$, we deduce that $M_{i,k}$ goes from $B_{(k-1)}$ to $B_{(k)}$.

Remark now that a normalising family of degree $k$ induces a normalising family of degree $k-1$, and that $N_k$ and $N_{k-1}$ coincide on $B_{(k-1)}$ so that in the sequel we will omit the subscript of $N_k$.

We define $\bar{N}$ on $\mathbb{K}[x]_{(k)}$ by induction on the degree, as follows: $\bar{N}(1) = 1$, and for any monomial $m$ in $\mathbb{K}[x]_{(k)}$, of the form $m = x_{i_0} m'$ with $1 \leq i_0 \leq n$, $m' \in \mathbb{K}[x]_{(k-1)}$,

$$\bar{N}(m) = N(x_{i_0} \bar{N}(m')).$$

This construction is valid since $\bar{N}(m') \in \langle B \rangle$ and $x_i \bar{N}(m') \in \langle B^+ \rangle$. Moreover if $m = x_{i_0} m' = x_{i_1} m''$ with $i_0 \neq i_1$, then we have $m = x_{i_0} x_{i_1} m'''$ and

$$N(x_{i_0} \bar{N}(m')) = N(x_{i_0} N(x_{i_1} \bar{N}(m'''))) = M_{i_0,k} \circ M_{i_1,k-1}(N(m'''))$$

By hypothesis, the two quantities are equal, so that the definition of $\bar{N}$ does not depend on the decomposition of $m$ as a product of a variable by a monomial.

We now check, also by induction, that $\bar{N}$ coincide with $N$ on $\langle B^+ \rangle$: $\bar{N}(1) = N(1) = 1$. For any monomial $m \neq 1$ in $B_{(k)}^+$, by the connexity property of $B$ and by definition of $B^+$, there exists $1 \leq i_0 \leq n$ and $m' \in B_{(k-1)}$, such that $m = x_{i_0} m'$. By induction we have $\bar{N}(m') = N(m') = m'$, which implies that

$$\bar{N}(m) = \bar{N}(x_{i_0} m') = N(x_{i_0} \bar{N}(m'))$$

We have to prove now that $\bar{N}(F_{(k)}) = 0$. By linearity, it is sufficient to prove that $\bar{N}(p) = 0$ for $p \in (f_j)_{(k)}, j = 1, \ldots, n$. But then, $p$ is of the form $p = x_{i_0} \cdots x_{i_d} f_j$ which implies that:

$$\bar{N}(p) = N(x_{i_0} \cdots N(x_{i_d} N(f_{j})))) = N(x_{i_0} N(x_{i_1} \cdots N(0) \cdots)) = 0.$$

Let us prove now the unicity of the extension $\bar{N}$ of the projection on $B_{(k)}$ such that $\bar{N}(F_{(k)}) = 0$. To do so, we prove that $\ker(\bar{N}) = F_{(k)}$. First we show that $\forall m \in \mathbb{K}[x]_{(k)}$, $m - \bar{N}(m) \in F_{(k)}$. We proceed by induction on the degree $k$. For $k = 0$, it is true since $\bar{N}(1) = 1$. Suppose that $\forall m' \in \mathbb{K}[x]_{(k-1)}$, $m' - \bar{N}(m') \in F_{(k-1)}$, and consider a monomial $m$, such that $deg(m) = k > 0$. Then there exists $x_{i_1}, m'$ s.t. $m = x_{i_1} m'$ and $m' - \bar{N}(m') \in F_{(k-1)}$. Therefore, $x_{i_1} m' - x_{i_1} \bar{N}(m') \in F_{(k)}$. Notice now that $x_{i_1} \bar{N}(m')$ has its support in $B_{(k)}$. From the definition of $\bar{N}$, there exists $q \in \langle F \rangle_{(k)}$ such that $\bar{N}(x_{i_1} \bar{N}(m')) = x_{i_1} \bar{N}(m') - q$. By substitution, we deduce that

$$m - \bar{N}(m) = x_{i_1} m' - \bar{N}(x_{i_1} \bar{N}(m')) \in F_{(k)}.$$ 

Thus, if $p \in \ker(\bar{N})_{(k)}$ then $p = p - \bar{N}(p) \in F_{(k)}$. Consequently, $\bar{N}$ is the projection onto $\langle B \rangle_{(k)}$ along $F_{(k)}$ which proves the unicity.

We directly deduce from this proof, the following result:

Corollary 1. With the hypothesis of theorem 1, we have $\mathbb{K}[x]_{(k)} = \langle B \rangle_{(k)} \oplus F_{(k)}$.

Definition 6. A choice function (resp. on $S \subset \mathbb{K}[x]$) refining the degree is a function $\gamma : \mathbb{K}[x] \to \mathbb{K}[x]$ (resp. $\gamma : S \to \mathbb{K}[x]$), such that for any polynomial $p \in \mathbb{K}[x]$ (resp. $p \in S$), $\gamma(p)$ is a monomial of the support of the maximal degree component of $p$.

Example 1. The degree lexicographical monomial order ($Dlex$) is a choice function that refines the degree. The function $\gamma$ of definition 2 is a choice function on the normalising family $F$. 
DEFINITION 7. Let \( \gamma \) be a choice function refining the degree. For an \( n \)-polynomials \( p_1, p_2 \), let \( C(p_1, p_2) \) be the C-polynomial relative to \( \gamma \) and \( (p_1, p_2) \) and defined by:

\[
C(p_1, p_2) = \frac{lcm(\gamma(p_1), \gamma(p_2))}{\gamma(p_1)} \cdot p_1 - \frac{lcm(\gamma(p_1), \gamma(p_2))}{\gamma(p_2)} \cdot p_2
\]

We will denote by the C-degree of \( (p_1, p_2) \), the degree of \( \frac{lcm(\gamma(p_1), \gamma(p_2))}{\gamma(p_1)} \cdot p_1 \), and by leading monomial of \( (p_1, p_2) \), the monomial \( lcm(\gamma(p_1), \gamma(p_2)) \).

It is almost the same definition as a \( S \)-polynomial (Cox et al. (1992)) when \( \gamma \) is a monomial ordering. We however need a new name to underline that now \( \gamma \) may not be such. It expresses commutation conditions between the \( M_{i,k} \). Indeed, if up to degree \( k \), the C-polynomials of a given normalising family of degree at least \( k \) vanish, then the \( M_{i,k} \) are pairwise commuting.

**Theorem 2.** Let \( k \in \mathbb{N} \), let \( F \) be a normalising family of degree \( k \) for the set \( B \) and let \( N_k \) be the induced projection from \( (B^+)^k \) onto \( (B)^k \). Then for any \( f, f' \in F_k \) such that \( C(f, f') \in (B^+)^{(k)} \). We suppose also that the \( \gamma(f) \) satisfies the hypotheses of theorem 1, i.e. \( \deg(\gamma(f)) = \deg(f) \). Then, \( N_k(C(f, f')) = 0 \) iff \( N_k \) extends uniquely as a projection \( N \) from \( \mathbb{K}[x]_k \) onto \( (B)^k \) such that \( \ker(N) = F_k \).

**Proof.** By theorem 1, we have to show that this condition is equivalent to the commutation of the operators \( (M_{i,d})_{d \leq k+1} \) on the monomials of \( B \). For any \( m \in B_{k+1} \) and any \( i \neq j \) such that \( x_i m \in B^+ \setminus B, x_j m \in B^+ \setminus B \), there exists \( f, f' \in F_k \) such that \( \gamma(f) = x_i m, \gamma(f') = x_j m \). Thus, we have

\[
M_{i,d+1}(M_{j,d}(m)) - M_{i,d+1}(M_{j,d}(m)) = M_{i,d+1}(\gamma(f) - f) - M_{i,d+1}(\gamma(f') - f)
\]

\[
= N(x_i \gamma(f) - x_i f) - N(x_j \gamma(f') - x_j f)
\]

\[
= N(x_i f' - x_i f) = N(C(f', f))
\]

which is zero by hypothesis. A similar proof applies if \( x_i m \in B \) or \( x_j m \in B \).

Conversely, since \( \ker(N) = F_k \) and \( C(f, f') \in F_k \), we have \( N(C(f', f)) = 0 \), which proves the equivalence and theorem 1.

### 3.2. The algorithm.
To be able to derive an algorithm from what we stated before (i.e., in order to have a normalizing family), we have to construct

- a monomial set \( B \) connected to 1,
- rewriting rules for all the monomials of \( B^+ \setminus B \) onto \( B \).

A rewriting rule in this case, is just a member of the ideal \( I \) generated by the given polynomials \( f_i \). Let us introduce a few notations:

- For any set of monomials \( M \) and any set of polynomials \( P, (M|P) \) denotes the matrix whose columns are indexed by the elements \( m \in M \), and rows by the polynomial \( p \in P \) and whose coefficients are the coefficients of the monomials \( m \in M \) in the polynomials \( p \in P \) (Mourrain and Trétuchet (2000)).
- ```newnum``` is a boolean whose value is ```true``` \( M_k \) if is non empty.

**Algorithm 1. Normal Forms**

**INPUT:** \( F = f_1, \ldots, f_m \) defining \( I \), ideal of dimension 0, and \( \gamma \) a choice function refining the degree.

**INITIALIZATION:** \( k = \min \{ \deg(f_i) \}, P_k = \{ f_i, \deg(f_i) = k \}, B = \bigcap_{i \in P_k} (\gamma(f_i))^e, M_k = \{ \gamma(f_i) \}. \)

**CORE LOOP:** While ```newnum|k < Maxdeg(F)+1``` do

- Compute \( C_{k+1} = \{ C(f, f') \) of degree \( k+1 \), \( f, f' \in P_k \}; \)
- Compute \( P_{k+1} = P_k^+ \cap (B^+); \)
- Compute \( E = \{ \proj_B(f) \mid \deg(f) = k+1 \}; \)
- if \( \exists f \in E, \deg(f) = k+1 \) then \( k = \min \{ \deg(f), f \in E \} - 1 \)
- \( P_{k+1} = P_{k+1} \cup \{ f \in \deg(f) = k+1 \}; \)
- \( P_{k+1} = P_{k+1} \cup \{ f \in E, \text{ s.t. } \deg(f) = k+1 \}; \)
- \( M_{k+1} = \{ M_k^+ \cap B^+ \}; \)
- \( F_{k+1} = \text{NormalisingFamily}(P_{k+1}, M_{k+1}); \)
• \( \text{Reduce } C_{k+1} \text{ with respect to } \cup_{j \leq k+1} P_j \);

• \( r = \#M_{k+1} - \#(P_{k+1}) \),
  \[ c = \#E, E = \{ \text{C-polynomials of } C_{k+1} \text{ non reduced to } \emptyset \} \cup \{ f \in (P_{k+1}) \backslash (F_{k+1}) \}, \]
  \( P_{k+1} = F_{k+1}, \)
  - \( r = 0 \) and \( c = 0 \): \( k = k+1 \).
  - \( r = 0 \) and \( c \neq 0 \): \( E' = \text{SelectMinDeg}_\gamma(E) \), remove from \( B \) the monomial ideal \( (\gamma(E')) \) and if \( f \in E' \) set \( k = \deg(f) \).
  - \( r \neq 0 \) and \( c \neq 0 \): \( \text{Increment}(B, P_{k+1}, M_{k+1}) \), apply \( \text{SelectMinDeg}_\gamma(E) \) to get another set \( E' \), remove from \( B \) the monomial ideal \( (\gamma(E')) \) and if \( E' \neq \emptyset \) set \( k = \min \{ f \in E' \mid \deg(f) \} \).
  - \( r \neq 0 \) and \( c = 0 \): \( \text{Increment}(B, P_{k+1}, M_{k+1}) \), and set \( k = k+1 \).

\textit{End WHILE}

\textbf{OUTPUT:} \( \{ P_j, j = 0..l \} \) a normalising family for all \( k \in \mathbb{N} \)

It uses the following subroutines:

• \( \text{NormalisingFamily}(P, M) \): performs linear combinations of the polynomials in \( P_{k+1} \) to put them in the form requested to be a normalising families, i.e. only one monomial outside of \( B \). More precisely,
  - it computes the matrix \( (P|M) \).
  - it multiplies \( P \), seen as a vector of polynomials, by a pseudo inverse of \( (P|M) \), which yields a temporary vector of polynomials \( V \).
  - it returns the polynomials set formed by the polynomials of \( V \) whose line correspond to a non zero line in the pseudo inverse of \( (P|M) \).

• \( \text{SelectMinDeg}_\gamma(E) \): applied to a polynomial set \( E \) is a function that implements to following algorithm:
  - Set \( d = \min(\deg(g), g \in E) \)
  - Construct the list \( E' = \{ f \in E ; \deg(f) = d \} \)
  - for \( f \in E' \) do
    * apply \( \gamma \) on \( f \) to get a monomial \( m \).
    * Perform linear combinations between the elements of \( E' \) to eliminate \( m \) of the support of the other elements of \( E' \).
      - if those linear combinations produce polynomials of degree less than \( d \) return \( \text{SelectMinDeg}_\gamma(E') \)
      - if no decreasing of the degree of the members of \( E' \) occurs, return \( E' \) together with the monomial set \( \gamma(E') \) constructed during the for loop.

Intuitively this function construct from the polynomial set \( E \) a set \( E' \) suitable to be added to the normalising family we are constructing.

• \( \text{proj}_B \) is the function that coincide with \( N_k \) on \( \mathbb{K}[x]_{(k)} \backslash B_{k+1}^+ \) and is the identity on \( B_{k+1}^+ \).

• \( \text{Increment}(B, F, M) \):
  - adds to \( B \) the monomials corresponding to the elements of \( M \backslash \gamma(F) \).
  - add to \( F \) the polynomials \( m = N(m) \), with \( m \) of degree \( k \) and such that \( \exists m', m'' \) monomials and variables \( x_{i_0}, x_{i_1} \) with \( m' \in M - \gamma(F), m'' \in B \) and \( m' = x_{i_0}^m m'' \) and \( m = x_{i_1}^m m'' \).

Intuitively this manipulation is needed to insure that we will always have in \( M_k \) all the monomials of \( B_{k+1}^+(k) \).

We will now prove that the above algorithm stops and produce a correct result. To do so we need the following lemmas.

\textbf{Lemma 1.} The variable \( k \) cannot take infinitely many times the same value \( d \).
Proof. First of all remark that there is finitely many cases where $k$ decreases below $d$, during this algorithm. Indeed each decreasing of $k$ under degree $d$, means that we have excluded from $B$ a monomial of degree less than $d$. As we never add to $B$ monomials of fewer degree than the current, and as there are finitely many monomials of a given degree; for all degree $d$ there exists some index $l_d$ such that for all subsequent loops of the algorithm after loop $l_d$ we have $k \geq d$.

Definition 8. Given a degree $d$, we will call $l_d$ the lowest integer such that after step $l_d$ the algorithm does not come back at degree $d$.

Definition 9. Let $I$ be an ideal of $\mathbb{K}[x]$. We will call the Hilbert function of $I$ the function over the nonnegative integers $s$ defined by:

$$HF_I(s) = \dim \mathbb{K}[x]_s/I(s)$$

Proposition 3. Given a degree $d$ and $l_d$ the index previously defined, the value of the Hilbert function of the ideal for this degree is the number of monomials in $B$ of degree less than $d$.

Proof. First of all, remark that the value of the Hilbert function cannot be greater than the number of monomials in $B_{(d)}$, because for all $m$ out of $B_{(d)}$ there exists a polynomial in $I$, whose support is included in $B_{(d)} \cup \{m\}$. Next suppose that $H_I(d) < \#B_{(d)}$. Then there exists a nonzero polynomial $p = \sum_{c \in J \subset \mathbb{N}^n} c_i^j f_i$, such that $\text{Supp}(p) \subset B_{(d)}$. Let us call $d'$ the maximal degree of the summands $x^{a_i} f_i$, and let us continue the algorithm until the step $l_d$. Then by theorem 1 we can define a projection $N$ over $\mathbb{K}[x]_{(d)}$ with $\ker(N) = (F)_{(d')}$, Consequently we have for each summand: $N(x^{a_i} f_i) = 0$, so that $p \in \ker(N) \cap \text{im}(N)$, i.e. $p = 0$ which is a contradiction.

Proposition 4. This algorithm stops and yields a normal from $N$ for the zero-dimensional ideal $I$ generated by the elements $f_1, \ldots, f_m$.

Proof. According to the previous proposition for each degree $d$ there exists an index $l_d$ such that the current degree cannot drop below $d$ for all subsequent loops in the algorithm. Moreover, $k \geq l_d$ implies $H_I(d) = \#B_{(d)}$. We now end the proof by noticing that as $I$ is zero dimensional, there exists a degree $d$ such that the Hilbert function is constant for all $d' \geq d$. That is to say, after the step $l_d$ the algorithm necessarily stops. When the algorithm stops, we have a normalizing family of degree $d$ for the set $B$. The remaining thing we have to check is to apply theorem 1, that is $B$ is connected to 1. To prove it, let us remark that if $B$ is connected to 1, it remains so when we increment it. Indeed a monomial added to $B$ is in $B^+$. Moreover if $B$ is connected to 1, and if we remove from $B$ some monomial ideal in the test phase, $B$ remain so. Finally, as $B$ is connected to 1 at the begin of the algorithm it remains so all along the computation. We have the commutation of the $M_i$’s because the C-polynomials all reduce to 0. So by theorem 1, we can define the projection $N_{i(d)}$ for all the monomials of degree $d$, and as there is no monomial of higher degree in $B$, $N_{i(d)} = N_{i(\infty)}$ and we are done.

We can notice here that there is far too many polynomials computed at each step, more precisely:

Proposition 5. If, up to degree $d$, the C-polynomials reduce to zero, then to compute the polynomials in $P_d$ from those in $P_{d-1}$, we only need to compute one polynomial per new monomial introduced (for each monomial in $M_{d}$, one polynomial whose leading monomial is the monomial in question).

Proof. What we need to prove is that if the C-polynomials reduce to 0, then any two new polynomials sharing the same leading monomial lead to the same rule. Let us take two distinct polynomials $p_1, p_2 \in P_{d-1}$ such that there exists two variables $x_1, x_2$ with $x_1 p_1$ and $x_2 p_2$ sharing the same leading monomial. Let $P = x_1 p_1 - x_2 p_2$. If $P$’s C-leading term is in $B^+$, there exists $q \in P_{d-2}$, such that $\gamma(p_1) = x_2 \gamma(q)$ and $\gamma(p_2) = x_1 \gamma(q)$. As we deal with step $d$, we have $p_1 - x_2 q$ and $p_2 - x_1 q$ both reduce to 0. We have now reduced our problem to showing that $r = x_1 N(x_2 q) - x_2 N(x_1 q)$ reduces to 0. Since $x_1 \neq x_2$ the polynomial $r$ has a support in $B^+$, and we checked during the algorithm that commutation occurred up to degree $d$.

Remark 1. We can also notice that only a small fraction of the C-polynomials computed are actually needed, namely those whose leading monomial is minimal for the division.

4. Experimentations
4.1. A framework for symbolic and numeric computations Our experimentations have been performed using the environment SYMPS\textsuperscript{2}. We are going to describe briefly this framework for symbolic and numeric computations. Its main objective is to provide, inside a single coherent platform, many of nowadays freely available softwares. The kernel of this platform is formed by the former ALP library (Mourrain (1996-1999)), this kernel provide a collection of parameterised but efficient classes of basic algebraic objects such as vectors, matrices, monomials and polynomials, ... which can be used easily in the construction of more elaborated algorithms. We have distinguished three levels of structure, implemented either as classes or namespaces: containers, views and modules. A special attention is paid to genericity or more precisely to parameterised classes (templates) which allow us to tune easily the data-structure to the problem to be solved. So called template expression are used to guide code expansion during the compilation process and thus to get most efficient implementations. We also consider the reusability of external libraries as one of the major issues of the development of the kernel. Currently, we have connected LAPACK (fortran library for numerical linear algebra), UMFPACK, SPARSELIB (respectively fortran and C++ libraries for sparse matrices), GMP (C library for extended arithmetics), GB, RS, ... Thus, specialised implementation for instance using LAPACK routines, can coexist with generic one, the choice being determined by the internal representation. We also want this library to be easy to use. A special attention has been paid to the connection to the SUPERLU library (Demmel et al. (1999)) (C library for solving sparse linear system). Indeed not only we connected SYMPS to SUPERLU but we realised a port of the routine of LU-decomposition in C++ in order to allows this routine to be parameterized toward the coefficient type, i.e. we are now able to reuse the very efficient algorithm of SUPERLU not only in double, single, complex double, and complex single precision but with any arithmetic type! Of course when dealing with those four types, it is still possible to use the true SUPERLU routine.

For the definition of multivariate polynomials, we have defined the monomials Monom\textlangle COEFFICIENTS, numerexp\textgreater. This type corresponds to monomials with coefficients left to the user’s choice and exponents stored as a number. The polynomials can then be defined, using the STL container vector by the type Poly\textlangle COEFFICIENTS, vector Monom\textrangle, Dlex. The class Dlex is a trait class for the implementation of the degree ordering refined by the lexicographic order on the monomials Mon. We could have used any other total order on the monomials. Notice that we could also have used here the list of the STL library, but it appeared to be less efficient in this problem.

The matrices St are based on the container sparse::rep2d\. They are of type MatSps\textlangle\textrangle, sparse::rep2d<\textlangle\textrangle, sparse::rep2d<double> > (or equivalently MatSps<double> ) which provides the vector space operations and the matrix-vector multiplication, for sparse matrices. The container type sparse::rep2d corresponds to the NCFORMAT type of SUPERLU so that no copy of objects are needed to call the external routines. In fact we have ported the LU decomposition routine of SUPERLU in C++ to be able to use generic coefficients.

Our experimentations have been performed on an PC workstation (PIII, 933 Mhz, 512 Mo of RAM)

4.2. The autocalibration problem This example corresponds to 6 quadratic equations of $P^5$, which come from an autocalibration problem. See (Faugeras (1993)) for more details. We consider two positions of the same camera, which is modelled by plane and a centre of projection (its optical centre). The images $m, m'$ of a same point satisfy the relation $m' F m = 0$ where $F$ is a $3 \times 3$ matrix, called the fundamental matrix associated to the two positions of the camera and which can be computed from points in correspondence in the image. The autocalibration problem consists in computing the intrinsic parameter matrix $A$ from the fundamental matrix $F$. This can be done by solving the so-called Kruppa equations (Kruppa (1913)),

$$F X F^t = \lambda^2 T_{e'} X T_{e'}$$

where $X = A A^t$ and $\lambda$ are unknown and $T_{e'}$ is the matrix of vectorial product by the epipole $e'$. Eliminating $\lambda$ in this equations yields 6 homogeneous quadratic equations in the 6 unknown coordinates of the $3 \times 3$ symmetric matrix $X$. The coefficients of these equations are approximated, for they come from measurements in the images. We have consider here 2 of these equations, compute the $25 = 32$ solutions (computed in 0.38s on alpha) and select the one which minimises the last polynomial. Here are the results of our experimentation.

\textsuperscript{2}http://www.inria.fr/galaad/logiciels/SYNAPS/
The module of the values of the polynomials \( f_1 \) at the computed root is bounded by \( 10^{-8} \). Using the matrix of multiplication \( M_0 \) instead of \( M_3 \) in the resolution leads to worse solutions. This phenomenon is due to the ill-conditioning of the problem with respect to this variable.

### 4.3. Blind identification

We consider the following signal processing problem and refer the reader to (Comon et al. (1998)) for more details. We want to transmit arrays \( x(n) \) of information depending on the (discrete) time \( n \) where \( x(n) \) is a vector of \( p \) numbers into a channel of length \( L \). These informations are received by \( k \) antennas which output a \( k \) array \( y(n) \) corresponding to the response signal. The impulse response of the system linking the sources with the antennas is given by a \( k \times p \) matrix \( H(n) \) so that we have the following relation:

\[
y(n) = \sum_{m=0}^{L-1} H(m)x(n-m) + b(n),
\]

where \( b(n) \) represents the noise. The problem consists in identifying the matrix of impulse response \( H \) from the observation \( y(n) \), without knowing the source signal \( x(n) \). This is not far from the model used for instance in mobile phones for the identification and follow-up of calls.

It is assumed that the sources are of a special type (i.e., cyclostationary) and \( b \) is a centred Gaussian noise, so that a statistic analysis of the output yields the following polynomial relations:

\[
\sum_{m=0}^{L-1} \sum_{i=1}^{p} h_{a,i}(m)h_{b,i}(m-l)(-1)^{n-m} = E(y_a(n)y_b(n-l)),
\]

where \( E \) stands for the mean. The quantities \( E(y_a(n)y_b(n-l)) \) are computed from the observed signal \( y(n) \) on a time period \( n_0, \ldots, n_1 \), and corresponds to approximate coefficients in our equations. Here is one of the 64 roots (computed in 0.4s on alpha) of our system:

<table>
<thead>
<tr>
<th>Root</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x0</td>
<td>-1.803468527372455</td>
</tr>
<tr>
<td>x1</td>
<td>-5.162835380624794</td>
</tr>
<tr>
<td>x2</td>
<td>-7.56575900599482</td>
</tr>
<tr>
<td>x3</td>
<td>-6.893354578266418</td>
</tr>
<tr>
<td>x4</td>
<td>-3.998807562745594</td>
</tr>
<tr>
<td>x5</td>
<td>-1.164422870375179</td>
</tr>
</tbody>
</table>

The module of the polynomials \( f_1 \) at this point is bounded by \( 10^{-8} \).

### REFERENCES


INRIA, GALAAD,
2004 route des Lucioles,
B.P. 93, 06902 Sophia Antipolis,
FirstName.Secondname@ sophia.inria.fr,
http://www.inria.fr/galaad/Secondname/

Received January 2002