J. Symbolic Computation (1994) **11**, 1–000

Efficient Computation of Zero–dimensional Gröbner Bases by Change of Ordering

J.C. FAUGÈRE[†], P. GIANNI[‡], D. LAZARD[†] and T. MORA[§]

[†]*LITP, Université Paris VI, Case 168, 4 place Jussieu, F-75252 Paris Cedex 05*
[‡]*Dipartimento di Matematica, Università, via Buonarroti 2, I–56127 Pisa*
[§]*Dipartimento di Matematica, Università, via L.B. Alberti 4, I-16132 Genova*

(Received 2 February 1994)

We present an efficient algorithm for the transformation of a Gröbner basis of a zero–dimensional ideal with respect to any given ordering into a Gröbner basis with respect to any other ordering. This algorithm is polynomial in the degree of the ideal. In particular the lexicographical Gröbner basis can be obtained by applying this algorithm after a total degree Gröbner basis computation: it is usually much faster to compute the basis this way than with a direct application of Buchberger's algorithm.

1. Introduction.

One of the main tools for solving algebraic systems is the computation of Gröbner bases (also called standard bases); we refer to Buchberger, (1965, 1970, 1979, 1985), Davenport et al. (1986) and Becker, Weispfenning (1993) for basic facts on this notion. The fact that the solutions come easily from the Gröbner basis for the lexicographical ordering appears in Trinks (1978). That the solutions of an algebraic system may be computed from the Gröbner basis for an other ordering on the monomials appears in Kobayashi et al. (1988); in Lazard (1989), several algorithms are given for computing the solutions from a Gröbner basis (depending on the ordering), together with a discussion on the meaning of solving an algebraic system.

The Gröbner basis of an algebraic system strongly depends on the choice of the ordering on the monomials. Different orderings have different advantages.

From a complexity point of view, the best ordering is the degree–reverse–lexicographical one; for this ordering, the computation of the Gröbner basis of a system of polynomial equations of degree d in n variables is polynomial in d^{n^2} if the solutions are finite in number (Caniglia et al. 1988 and 1991); this complexity decreases to d^n if the solutions at infinity are also finite in number (Lazard, 1983). In practice the computations are generally much faster and much more feasible than with other orderings.

The pure lexicographical ordering leads to computations which are much longer than with degree orderings and even are often untractable; the corresponding complexity has been proved to be $d^{O(n^3)}$ when the number of solutions is finite (Caniglia et al. 1988); our algorithm reduces this complexity to $d^{O(n^2)}$. For a practical point of view, the basis

0747–7171/90/000000 + 00 $03.00/0

given by this ordering is better suited for computing the solutions (Trinks, 1978; Gianni, Mora, 1987; Lazard, 1989).

There are many other orderings between these extreme cases; especially there are the elimination ones which appear in Bayer–Stillman's Macaulay system.

Thus, an efficient algorithm for change of ordering is useful. We give such an algorithm in the zero-dimensional case (finite number of solutions).

This algorithm is implemented in Axiom and in the experimental software Gb developed by one of us. Examples show that, for obtaining the basis for the lexicographical ordering, it is usually much faster to compute the basis for the degree ordering and to use our algorithm than a direct computation by Buchberger's algorithm. In several examples we get the basis for the lexicographical ordering where previous methods fail, even on supercomputers.

Since the preliminary version of this paper (july 1989) the algorithms for computing Gröbner bases have made substantial progresses, especially owing to Sugar strategy (Giovini et al. 1991). It follows that some Gröbner bases become easy to compute and our algorithm is no more useful for them. However, much more Gröbner bases may now be computed and our algorithm is, in many cases, the only way for computing the lexicographical one.

It should be also quoted here that the matrices which are computed by our procedure *Matphi* appear now to be a fundamental tool for many aspects of solving process, especially numerical solving (Auzinger and Stetter, 1988; Möller,1993).

The fact that, in our algorithm for changing the ordering, the old ordering may appear only throughout these matrices, has another important application: our algorithm may be used for computing the Gröbner base after a linear or polynomial change of variables (Gianni, Mora, 1987).

2. Definitions.

In this paper we will denote by K a field, by $R = K[x_1, \ldots, x_n]$ the ring of polynomials in n variables with coefficients in K.

We will consider an ideal in R given by its Gröbner basis G with respect to some admissible ordering. We will say that an element $f \in R$ is *reduced* by G (or in *normal form* with respect to G) if no element $g \in G$ has leading term that divides any term of f; we will call *reduction algorithm* the algorithm that computes the normal form of a given polynomial f. A Gröbner basis is reduced if each of its elements is reduced by the others.

We will consider a zero–dimensional ideal, i.e. an ideal I such that the set of common zeros of the polynomials in I is finite in the algebraic closure of the field of coefficients; this is equivalent to the fact that, for each variable x_i, there is a polynomial in the Gröbner basis for I, with a power of x_i as a leading monomial, (Gianni et al., 1988).

DEFINITION 2.1. *Given a zero dimensional ideal I in R and $(G, <)$ a reduced Gröbner basis for I, we will call* the natural basis determined by G *of the K–vector space R/I, the basis* $\mathrm{B}(G)$ *whose elements are the reduced monomials with respect to G. We will denote by* $\mathrm{D}(I)$ *the dimension of the K–vector space R/I (the* degree *of the ideal I).*

We will use the properties of the structure of vector space of R/I; for this reason we want to analize a little closer the structure of the quotient ring and of the monomials that generate it.

DEFINITION 2.2. *Let* $\mathrm{B}(G)$ *be the natural basis for* R/I*, let*

$$\mathrm{M}(G) = \{x_i b \mid b \in \mathrm{B}(G),\ \ 1 \leq i \leq n,\ \ x_i b \notin \mathrm{B}(G)\}$$

be the bordering of G*.*

The following proposition characterizes the elements of $\mathrm{M}(G)$.

PROPOSITION 2.1. *Let* I *be a zero dimensional ideal,* $(G, <)$ *be the reduced Gröbner basis with respect to an admissible ordering* $<$*, and* $\mathrm{B}(G)$ *be the natural basis of* R/I*, then for every element* $m \in \mathrm{M}(G)$ *exactly one of the following conditions holds:*

(i) For each x_i *dividing* m*, we have* $m/x_i \in \mathrm{B}(G)$*; this is the case iff* m *is the leading monomial of an element of* G*.*

(ii) $m = x_j m_k$ *for some* j *and some* $m_k \in \mathrm{M}(G)$*.*

PROOF. *(i)*: This follows immediatly from the definitions of reduced Gröbner basis and of $\mathrm{B}(G)$.

(ii): Let x_j such that x_j divides m and $m/x_j \notin \mathrm{B}(G)$; then $m_k = m/x_j \in \mathrm{M}(G)$. In fact from $m = x_j m_k = x_i b$ we have $i \neq j$ and $m_k/x_i = b/x_j$ is in $\mathrm{B}(G)$, because $\mathrm{B}(G)$ is closed under division, by definition; thus $m_k = x_i(b/x_j) \in \mathrm{M}(G)$. □

COROLLARY 2.1. *Let* k *be the number of generators of a reduced Gröbner basis for a zero dimensional ideal* I*; then* $k \leq n\mathrm{D}(I)$*.*

3. Computation of Normal Form

When we work in the vector space R/I and we consider the natural basis determined by a given Gröbner basis G, in order to find the coordinates of an element, we have to compute its normal form with respect to G. This operation, as we remarked, can be obtained by straight application of the Buchberger's algorithm, but in this way we can not estimate well the complexity of this step. For this reason we will take advantage of the structure of vector space in order to construct an algorithm that will find the coordinate of normal forms of elements of R/I in polynomial time : we will consider the n-linear maps ϕ_i defined on the basis $\mathrm{B}(G)$ by

$$\phi_i : m \to NormalForm(x_i m)$$

and we will study their properties. We remark that for every element $b \in \mathrm{B}(G)$ and for every i, either $x_i b \in \mathrm{B}(G)$ or $x_i b \in \mathrm{M}(G)$, the bordering of G, defined in the previous section.

DEFINITION 3.1. *Let* I*,* $(G,<)$ *and* $\mathrm{B}(G)$ *be as in Proposition* **??**. *We define* $\mathrm{T}(G) = (t_{ijk})$ *as the* $n \times \mathrm{D}(I) \times D(I)$ *tensor whose elements are:*

$t_{ijk} =$ *j–th coordinate w.r.t.* $\mathrm{B}(G)$ *of the reduction by* G *of the element* $x_i b_k$ *(*$b_k \in \mathrm{B}(G)$*).*

The first result we obtain is :

PROPOSITION 3.1. *In order to compute* $\mathrm{T}(G)$*,* $O(n\mathrm{D}(I)^3)$ *arithmetic operations are sufficient.*

PROOF. Consider $\mathrm{MB}(G) = \mathrm{B}(G) \bigcup \mathrm{M}(G)$ and order its elements with respect to $<$. We will construct columns t_{i*k} by following the order in which the $x_i b_k$ appear in $\mathrm{MB}(G)$. Consider $m = x_i b_k$. If $m \in \mathrm{B}(G)$ then m is not reducible by G and so $t_{ijk} = 0$ for $j \neq k$ and $t_{ikk} = 1$. Otherwise $m \in \mathrm{M}(G)$ and so, by Proposition **??**, either m is the leading term of an element in G, $g = m + \sum_{u=1}^{\mathrm{D}(I)} a_u b_u$, and in this case $t_{i*k} = (-a_1, \ldots, -a_{\mathrm{D}(I)})^t$ or $m = x_l m'$ with $m' \in \mathrm{M}(G)$ and $m' < m$. In this latter case, the coordinates of $m' = x_s b_h$ w.r.t. $\mathrm{B}(G)$, have already been computed and are stored in t_{s*h}; so, in order to compute the t_{i*k}, it is enough to add the coordinates (already computed) of the products $x_l b_v$, $(b_v \in \mathrm{B}(G))$, multiplied by the corresponding coefficients, i.e. $x_i b_k = x_l x_s b_h = x_l \sum_v t_{svh} b_v = \sum_u \sum_v t_{svh} t_{luv} b_u$. In this way we have to perform $\mathrm{D}(I)^2$ operations in order to compute t_{i*k} and the result follows, since this has to be done at most $n\mathrm{D}(I)$ times. □

REMARK 3.1. In order to compute $\mathrm{T}(G)$ it is necessary to order the monomials in $\mathrm{MB}(G)$. For this purpose we can define a function *NextMonom* that sequentially "generates" the following monomial to consider. We shall discuss later about such a function. In any case this function doesn't involve any arithmetic operation.

REMARK 3.2. For $i = 1, \ldots, n$, the matrix associated to ϕ_i with respect to $\mathrm{B}(G)$ is t_{i**}.

We give now an algorithm that implements the construction described in the previous proposition. As we remarked the columns of the matrices t_{i**} can be used to compute the normal form of any element of the form $x_i p$ for a reduced polynomial p.

PROCEDURE 3.1. **Matphi**

Input :
$<$ an admissible ordering.
Basis, a minimal reduced Gröbner basis for a zero–dimensional ideal.

Output :
$\phi[i, m, m']$ for $i = 1, \ldots, n$ and for $m, m' \in \mathrm{B}(G)$, such that $\phi[i, *, *]$ is the matrix of the application $p \to \mathit{NormalForm}(x_i p)$ for p a reduced polynomial.

Subfunctions :
NextMonom removes the first element of *ListOfNexts* and returns it; returns nil if the list is empty.
InsertNexts(*monom*) adds to *ListOfNexts* the products of *monom* by all variables, sorts this list by increasing ordering for $<$ and remove duplicates.

Local variables :
ListOfNexts, the list of "next" monomials to be considered sorted by increasing ordering for $<$.

Begin
monom := 1;
ListOfNexts := [ ];
while monom $\neq$ *nil do*
if monom is a strict multiple of the leading term of some element of *Basis* *then*
let *monom* $= x_j m$ with m reducible w.r.t. *Basis*;
⟦the test being true we have *monom* $= x_j m$ and $m <$ *monom*; thus *NormalForm*$[m] = \sum \lambda_i m_i$ has been previously computed with $m_i \in \mathrm{B}(G)$ and $m_i < m$⟧

NormalForm[*monom*] := $\sum \lambda_i$ *NormalForm*[$x_j m_i$];
[[$x_j m_i <$ *monom*, thus *NormalForm*[$x_j m_i$] has been previously computed]]
for each k such that *monom* = $x_k m'$ with m' irreducible by *Basis*
do $\phi[k, m'', m']$:= coefficient of m'' in *NormalForm*[*monom*]
else if *monom* is the leading term of some element p of *Basis*
then
NormalForm[*monom*] := $-rest(p)$; [[(i.e. $p - leadingTerm(p)$]]
for each j such that *monom* = $x_j m$
do $\phi[j, m', m]$:= coefficient of m' in *NormalForm*[*monom*]
else
NormalForm[*monom*] := *monom*;
InsertNexts(monom);
for each j such that *monom* = $x_j m$

$$do \quad \phi[j, m', m] := \begin{cases} 1 & \text{if } m' = monom \\ 0 & \text{otherwise} \end{cases}$$

monom := *NextMonom*
end.

The correctness of this algorithm follows essentially from the proof of Proposition **??**. The algorithm distinguishes the same three cases for the monomials to be considered: the elements of *Basis*, that are irreducible, those which appear in the leading term of some element of the Gröbner basis, and the others. Only for the last group, the normal form is not immediate. It is computed using the fact that the monomial has the form $x_i m$, where the normal form of m has been computed, and that the part of the matrix ϕ_i, needed to multiply it by x_i has also been computed. Let us also remark that the first test (divisibility by some leading term) does not need any searching: for testing it suffices to count the number of insertions in *ListOfNexts*, the test returns true if this number is less than the number of variables explicitly appearing in it; for getting the decompositions *monom* = $x_i m$, with irreducible or reducible m it suffices to remember from which monomials *monom* was inserted. The second test (being the leading monomial) also does not need any searching: if *Basis* is sorted by increasing leading monomial, the only leading monomial which may be equal to *monom* is the first one which has not yet be equated to it.

4. Change of Ordering

We analyze in this section the main algorithm of this paper, the algorithm for the change of ordering.

PROPOSITION 4.1. *Let I be a zero dimensional ideal and $(G_1, <_1)$, the reduced Gröbner basis with respect to an admissible ordering $<_1$. Given a different ordering $<_2$, it is possible to construct the Gröbner basis $(G_2, <_2)$ with respect to the ordering $<_2$ with $O(n\mathrm{D}(I)^3)$ arithmetic operations.*

PROOF. From $(G_1, <_1)$ we can construct $\mathrm{B}(G_1) = \{a_1, \ldots, a_{\mathrm{D}(I)}\}$, $\mathrm{M}(G_1)$ and $\mathrm{T}(G_1)$ as in the previous section. We want to find the elements of $\mathrm{B}(G_2)$ and $(G_2, <_2)$. For this

reason we will construct a matrix C that will contain in the i–th column the coordinates of each element $b_i \in \mathrm{B}(G_2)$, with respect to $\mathrm{B}(G_1)$.

We start with $\mathrm{B}(G_2) := \{1\}$ and $\mathrm{M}(G_2) := \emptyset$ for constructing the new base iteratively (the polynomial 1 is certainly in $\mathrm{B}(G_2)$). We consider $m = min_{>_2}\{x_j b_i \mid 1 \leq j \leq n, b_i \in \mathrm{B}(G_2), x_j b_i \notin \mathrm{B}(G_2) \cup \mathrm{M}(G_2)\}$. Three cases can arise:

1. $m =$ leading term g, for some g to be inserted in G_2
2. m has to be inserted in $\mathrm{B}(G_2)$
3. m has to be inserted in $\mathrm{M}(G_2)$, but m is a strict multiple of the leading term of some g in G_2.

We can easily check if the third case holds: the leading term of g is strictly less than m for any admissible ordering and has already been inserted in $\mathrm{M}(G_2)$.

So we are left to consider case 1 or 2. Since, by construction, $m = x_j b_i$ we can compute its coordinates $c(m)_h$ with respect to $\mathrm{B}(G_1)$ by using the table C (so far computed) and $\mathrm{T}(G_1) = (t_{ijk})$:

$$m = x_j b_i = x_j \sum_k c_{ki} a_k = \sum_k c_{ki}(x_j a_k) = \sum_k c_{ki} \sum_h t_{jhk} a_h =$$

$$= \sum_h (\sum_k t_{jhk} c_{ki}) a_h = \sum_h c(m)_h a_h.$$

At this point, if the vector $c(m)$ is independent from the vectors in C, we are in the case 2 and we have found a new monomial $m \in \mathrm{B}(G_2)$; otherwise the dependency relation furnishes a new element $g \in G_2$.

It is easy to see that the whole construction needs $O((n)\mathrm{D}(I)^3)$ operations. In fact the computation of $c(m)$ involves only the product of a matrix by a vector of size $\mathrm{D}(I)$. If an echelon form of the matrix C is maintained, for testing linear independency and incrementing C, only $O(\mathrm{D}(I)^2)$ are needed. □

We describe now an algorithm that implements the construction of the previous proposition. We will call **NewBasis** the procedure that transforms a Gröbner basis of a zero dimensional ideal with respect to some given ordering into a new one with respect to a new ordering. In this algorithm we will call *NormalForm* the function that returns the reduced form of a polynomial with respect to the given basis, without specifying the algorithm used for this purpose. We will discuss again later about this function.

PROCEDURE 4.1. **NewBasis**

Input :

$<$ a new admissible ordering.

oldBasis, a Gröbner basis of a zero–dimensional ideal, with respect to some ordering.

Output :

newBasis, the reduced Gröbner basis of the ideal generated by *oldBasis* with respect to the ordering $<$.

Subfunctions :

NormalForm(polynom), returns the reduced form of a polynomial with respect to *oldBasis.*

NextMonom, removes the first element of *ListOfNexts* and returns it, returns *nil* if the list is empty.

InsertNexts(monom), adds to *ListOfNexts* the products of *monom* by all variables, sorts this list by increasing ordering for $<$ and remove duplicates.

Local variables :

staircase, the list of leading monomials of the elements of *newBasis*;

MBasis, a list of pairs $[a_i, b_i]$ where $[\ldots, a_i, \ldots]$ is the list of monomials which are in normal form with respect to *newBasis* and $b_i = NormalForm(a_i)$, the normal form of a_i with respect to $oldBasis$. We select elements from each pair with selectors *first* and *second*;

ListOfNexts, the list of "next" monomials to be considered sorted by increasing ordering for $<$.

Begin

MBasis := [ ]; *staircase* := [ ]; *newBasis* := [ ]; *ListOfNexts* := [ ];
monom := 1;
while monom $\neq$ *nil do*
if monom is not a multiple of some element of *staircase*
⟦We are in case (1) or (2) of the proof of Proposition **??**⟧
then
vector := *NormalForm*(*monom*);
if there exist a linear relation: ⟦case(1)⟧
$vector + \sum_{v \in MBasis} \lambda_v second(v) = 0$
then
$pol := monom + \sum_{v \in MBasis} \lambda_v first(v)$;
newBasis := *cons*(*pol, newBasis*);
staircase := *cons*(*monom, staircase*);
else ⟦case (2)⟧
MBasis := *cons*([*monom*, *vector*], *MBasis*);
InsertNexts(*monom*);
monom := *NextMonom*
end Newbasis.

Correctness of the algorithm: Suppose, for the moment, that the main loop is finite. We first prove that the elements of *MBasis* are linearly independent modulo the ideal generated by *oldBasis*: if this were false, there would be a linear combination $P = \sum \lambda_i m_i$ of elements of *MBasis* which is a polynomial in ideal(*oldBasis*). Then

$$NormalForm(P) = \sum \lambda_i NormalForm(m_i) = 0$$

and this gives a relation which implies that the algorithm would put in *staircase* rather than in *MBasis* the greatest of the m_i such that $\lambda_i \neq 0$.

The elements of *newBasis* are in ideal(*oldBasis*): if

$$P = monom + \sum \lambda_v m_v$$

is such an element, then

$$NormalForm(P) = NormalForm(monom) + \sum \lambda_v NormalForm(m_v) = 0$$

by construction of P.

Finally the elements of *staircase* are the leading monomials of the elements of *newBasis*: this is clear because the loop works with increasing monomials.

Now, each monomial which is not in *MBasis* nor in *staircase* is multiple of some element of *staircase*. This is clear if the monomial appeared in *ListOfNexts*; otherwise suppose that the monomial is of the form $m_1 m_2$ where m_1 is in *MBasis* and maximal for this property. In this case the monomial, if not in *MBasis*, is multiple of some element which appeared in *ListOfNexts* and was not put in *MBasis*; it has been put in *staircase* or is a multiple of some element of *staircase*. Thus, the normal form of a polynomial with respect to *newBasis* is a linear combination of elements of *MBasis* and the normal form of an element of ideal(*oldBasis*) is zero (first assertion above).

This proves that *newBasis* is a Gröbner basis for the new ordering; it is minimal and reduced because, by construction, none of its monomials is a multiple of a leading monomial other than itself.

We have now to prove that the algorithm *NewBasis* stops; we are in the zero dimensional case; so the maximal number of linearly independent irreducible polynomials is the (finite) dimension of the quotient of the polynomial ring by the ideal, i.e. the number of irreducible monomials for any Gröbner Basis. Thus the number of iterations which increase *MBasis* is finite. When *MBasis* is complete *ListOfNexts* may no more increase, thus the number of remaining iterations is bounded.

Management of NextMonom: When inserting monomials in *ListOfNexts*, it is useful to remember that it is obtained as the product of a variable by a monomial with known normal form; this is easy to implement with pointers. The same monomial may be obtained several times. When removing duplicates, it is useful to store in a counter the number on insertions of the monomial in *ListOfNexts*; this make the test *if monom is a multiple of some element of staircase* very easy: the quotient of an element of *staircase* or *MBasis* by any variable is necessary in *MBasis*, but this is false for the other monomials (see Proposition **??**); thus the test may be replaced by *if the number of insertions of monom in ListOfNexts is greater than the number of variables in monom*, which is much faster.

To conclude this section we want to make few remarks on the function *NormalForm*. At first, let us remark that *NormalForm* is called on monomials of the form $m = x_i m'$ where m' is in *MBasis* and its normal form has been computed and stored in *vectBasis*. Thus, if p is the normal form of m', the normal form of m may be obtained by

$$NormalForm(m) = NormalForm(NormalForm(x_i) \times p).$$

It is much more efficient to apply the reduction process to $x_i p$ than to m, because the first one may be obtained from the second one by a partial reduction. The extra cost for finding x_i and p is very low: it suffices, in the procedure *InsertNexts*(m), to store in *ListOfNexts*, with $x_i m$, the variable x_i and a pointer to the normal form of m (which is the actual value of *vector*).

In summary, as we have shown in section 2 we could use the procedure **Matphi** and in this way obtain a polynomial time algorithm, but the method described in the last paragraph seems to be the best in practice.

5. Complexity; case of fields with unit cost.

In this section we will summarize all the results on complexity we obtained in the previous sections. The results are rather different if the operations in the field of coefficients take a constant time, or if we take into account the growth of the coefficients. In this section, we consider only the first case.

As the size of a multivariate polynomial may exponentially depend on its representa-

tion, we need a convenient measure of the size of the input. It appears that two parameters are important:

the number n of variables;

the number $D = D(I)$ of monomials which are in normal form; it is the number of elements in the natural basis $\mathrm{B}(G)$ and does not depends on the ordering, being also the number of common zeros (with multiplicity) of the polynomials in the ideal generated by *oldBasis* and the dimension as a vector space of the quotient of the polynomial ring by this ideal.

We will prove that *NewBasis* is polynomial in D and n for fields with unit cost. This implies that *NewBasis* is a polynomial algorithm for a natural representation of data, which we define now.

As a polynomial in normal form with respect to *oldBasis* has at most D terms, we may consider a dense representation for such polynomials: the list of coefficients of the irreducibles monomials. With this representation, D is also the size of a polynomial in normal form.

Now, the size of the input for this representation is $nD + k(n + D)$, where k is the number of polynomials in *oldBasis*: the list of the irreducible monomials has size nD and each polynomial in the basis is represented by its leading monomial (size n) and its reductum (size D).

By corollary **??**, k is at most nD; it is at least n for a zero–dimensional ideal; thus:

PROPOSITION 5.1. *If the size of the elements of the field of coefficients is* 1*, then, with the above representation, the size S of the input of NewBasis satisfies*

$$2nD + n^2 \leq S \leq nD^2 + n^2D + nD.$$

REMARK 5.1. In practice, except when D is very small, the number of elements of a Gröbner basis is less than D, and $S \leq D^2 + 2nD$.

We are now ready to state and prove our result of complexity.

THEOREM 5.1. *If the basis field operations need an unit time, algorithm NewBasis implemented with Matphi needs $O(nD^3)$ field operations; the rest of the computation, essentially the monomial manipulations, needs a total time of $O(n^2D^2)$ with an elementary implementation of ListOfNexts or of $O(n^2D\log(nD))$ if this list is implemented as an efficient priority queue.*

For the above representation of the input data, algorithm NewBasis is polynomial, on a field with unit cost for operations.

PROOF. Consider first the complexity of the monomial operations; as quoted above, the tests do not need searching, but only need to compare monomials, which needs a time of $O(n)$; thus, the main time is devoted to the management of *ListOfNexts*; this list has a length of at most nD; if it is sorted, *InsertNexts* is a merge with a list of length n, and this merge needs $O(nD)$ comparisons of monomials (time $O(n)$); this gives a total time of $O(n * nD * D)$, *InsertNexts* being called D times.

It may be remarked that *ListOfNexts* may be viewed as a priority queue with nD insertions and nD deletions of the least element; methods for implementing priority queues are well known, which need a total time proportional to the number of insertions/deletions

times the logarithm of the maximal length of the list; this has to be multiplyed by the time of the comparisons, giving the improved result in the theorem.

Let us compute now the number of field operations. In *Matphi* we have to compute at each non trivial iteration, a linear combination of D vectors of dimension D, which needs D^2 field operations. Thus *Matphi* needs $O(nD^3)$ field operations. The linear algebra computations in *NewBasis*, in the whole, is equivalent to triangularize a $D \times nD$ matrix, thus it needs also a time of $O(nD^3)$, which finishes the proof of the theorem. □

6. Complexity in bit operation with growth of coefficients.

If we try to take into account the growth of the coefficients, it appears that it is *Matphi* which introduces a growth which is exponential in nD; we have seen that *NewBasis* may be computed in a time which is exponential in n, but polynomial in D. We conjecture that this exponential behaviour is unavoidable. This conjecture is supported by looking at the size of coefficients in example (V) of next section.

For the complexity analysys in this section we restrict ourselves to the case where the old ordering is a degree ordering (i.e. $degree(m) < degree(n)$ implies $m < n$ for monomials m and n); this is not a strong restriction, because Buchberger's algorithm is unefficient with other orderings, and, thus, *NewBasis* is mainly useful for changing from a degree ordering to another ordering.

For taking into account the growth of coefficients, the use of *Matphi* is no longer convenient; thus, in this section we will use the first optimized implementation of *NormalForm* described in Section 4; that is, we will apply the standard reduction process only on the product of a polynom in normal form by a variable.

We introduce a new measure of the size of the problem, E that is the number of monomials that are not greater (for the old ordering) than any monomial of the form $x_i m$ for some irreducible monomial and some variable. Thus E is the maximal number of different monomials which may appear in the normal form computation in *NewBasis*.

Before proving that *NewBasis* is polynomial in n and E, it is useful to give estimates for E.

PROPOSITION 6.1. *For a degree ordering, we have*

$$D \leq E \leq \begin{pmatrix} n+D \\ n \end{pmatrix} = \frac{(n+D)!}{n!D!} \leq D^n.$$

If the maximal degree of the irreducible monomials for the (old) basis is d then

$$E \leq \begin{pmatrix} n+d+1 \\ n \end{pmatrix} = \frac{(n+d+1)!}{n!(d+1)!}.$$

The monomials counted by E are of degree at most D or $d+1$; thus, the number of such monomials is at most the number of monomials of degree at most D or $d+1$.

REMARK 6.1. In most cases, for degree–reverse–lexicographical ordering, there are few of the monomials counted by E which are not of the form $x_i m$ with m reduced; thus E is not much greater than D; thus the exponential behaviour appears only with very special (non generic) examples.

We are now able to compute the complexity of *NewBasis*; taking into account the

growth of coefficients introduces a difference only in the normal form and linear algebra computations; it has been quoted above that the linear algebra part is equivalent to the triangularization of a $D \times B$ matrix where $B \leq E$ is the number of iterations of the main loop of *NewBasis*; thus, the linear algebra part is polynomial, provided that the reduction of a $E \times E$ matrix is a polynomial problem over the field of coefficients.

In the computation of a normal form, a step of reduction of a polynomial p by an element f of the Gröbner basis consists in replacing p by some linear combination $p - cmf$ where c is a coefficient and m a monomial; each polynomial which appears may be viewed as the vector of its coefficients; this vector is of length E, by definition of E; it is easy to see that the vector representing $p - cmf$ is the product of the vector representing p by a triangular matrix depending only on m and f, because c is such that the leading term of cmf is the same as the corresponding term of f; the coefficients of this matrix are 0, 1 or the quotients of the coefficients of f by its leading coefficient. The number of steps of reduction for computing the normal form is at most E, the number of monomials which may be reduced. Thus a call to *NormalForm* corresponds to a product by E matrices of a vector which is a shifted result of *NormalForm* or represents a monomial; *NormalForm* being called at most E times its result is a column of the product of at most E^2 matrices with the coefficients of the initial Gröbner basis as entries.

Thus, the part of *NewBasis* dealing with coefficients consists in computing some products of matrices and triangularizing the resulting matrix. If both parts are polynomial problems, the whole is a polynomial algorithm. Before stating this, we need a definition.

DEFINITION 6.1. *A field is* polynomial, *if there exist polynomial algorithms for solving the following problems:*

Triangularize a $k \times k$ matrix with coefficients in the field.

Compute the product of k matrices $k \times k$.

PROPOSITION 6.2. *The following fields are polynomial: Finite fields, rational number field, multivariate rational fractions over the preceedings (with dense representation).*

For finite fields, it is easy. The other fields are quotient fields; thus we may choose a common denominator at the beginning and work on the integers or on a polynomial ring over the integers. It suffices to show that both problems need a polynomial number of operations and that the field elements which appear during the computations have a polynomial size.

For the matrix product problem, the element which appear are, at most, the sum of k^{k-1} products of k terms; for the triangularization problem, with Bareiss algorithm [BAR], the coefficients which appear are determinants, thus the sum consists of at most $k!$ products of k terms; it is easy to verify that this produces results of polynomial size, proving the proposition.

THEOREM 6.1. *NewBasis, with the first optimisation described in Section* 4 *is polynomial in E over polynomial fields.*

REMARK 6.2. We leave to the reader the determination of the degree of the complexity of *NewBasis* over an explicit field, such as the rational numbers; this is easy, but not very important: it follows from the proof and from practical experiments (see below) that the time of the computations depends mainly on the size of the results. Thus prohibitive

computations may only appear when the result is "too big to be useful", and the real asymptotic behaviour does not appear in practical computations. In other words, on most entries, the algorithm is polynomial of low degree with respect to the size of *the output.*

Finally, we may apply the previous results of complexity to the whole computation of the Gröbner basis.

THEOREM 6.2. *Let I be a zero–dimensional ideal of n–variate polynomials over a polynomial field, generated by polynomials of degree at most d. If the generators of I have only a finite number of common zeros at infinity, then, its Gröbner Basis for any ordering may be computed in a time polynomial in d^n. If the set of zeros at infinity is not finite, then the same conclusion is true for a time which is polynomial in d^{n^2}.*

PROOF. If the set of zeros at infinity is finite, Lazard, 1983, has proved that the Gröbner basis for the degree–reverse–lexicographical ordering may be computed in time polynomial in d^n, and contains polynomials of degree at most $nd-n+1$; it follows immediately that the monomials counted by E are at most of this degree and that $E \leq e^n d^n$ where e is the basis of naturals logarithms. Thus applying $NewBasis$ on this basis needs also a time which is polynomial in d^n.

If the ideal is zero–dimensional but not necessarily at infinity, Caniglia et al., 1991 have shown that one may compute, in time $d^{O(n^2)}$, a new set of generators which are of degree $d^{O(n)}$ and have no zeros at infinity. With this new set of generators, the second part of the theorem follows from the first one. □

7. Examples.

In this section, we define some typical example and list various computer times achieved on them as well some related parameters such that the number of solutions or the number of digits in the highest coefficient of the output.

We give two timing tables: the first one appeared in the first version of this paper and correspond to the state of the art in 1988. The computation were done on a computer IBM 4381 (3.5 to 4 Mips). The second one correspond to present state of the art (1993): The computer is now a SUN Sparc 10, and Gröbner base computations are done using Sugar strategy (Giovini et al., 1991) (or, for the cyclic 7-th roots, a recent algorithm not yet published); the Gröbner base implementation as well as the change base ordering are those of the software GB developed by J.C. Faugère.

7.1. CYCLIC n-TH ROOTS

In the group $\mathcal{S}_n$ of permutations of $\{1,\ldots,n\}$, let us consider σ_0 the cycle $(1,2,\ldots,n)$ and G_n the cyclic subgroup generated by σ_0.

The cyclic n-th roots system is defined by:

$$C(n) \quad \begin{cases} x_1 + x_2 + \cdots + x_n, \\ \sum_{\sigma\in G_n} x_{\sigma(1)}x_{\sigma(2)}, \\ \sum_{\sigma\in G_n} x_{\sigma(1)}x_{\sigma(2)}x_{\sigma(3)}, \\ \cdots\cdots\cdots, \\ \sum_{\sigma\in G_n} x_{\sigma(1)}\cdots x_{\sigma(n-1)}, \\ x_1\cdots x_n - 1 \end{cases}$$

7.2. Auxiliary system in cyclic 7-th roots

This is the cyclic 7-th roots system in which we made the following change of variables:

$$x_1 = 1, \quad x_2 = a, \quad x_3 = b, \quad x_4 = c, \quad x_5 = \frac{1}{a}, \quad x_6 = \frac{1}{b}, \quad x_7 = \frac{1}{c}$$

This lead to the system:

$$\text{Aux} \quad \begin{cases} a^2bc + ab^2c + abc^2 + abc + ab + ac + bc, \\ a^2b^2c + ab^2c^2 + a^2bc + abc + bc + a + c, \\ a^2b^2c^2 + a^2b^2c + ab^2c + abc + ac + c + 1 \end{cases}$$

7.3. Caprasse's system

It is the system:

$$\text{Cap} \quad \begin{cases} y^2z + 2xyt - 2x - z, \\ -x^3z + 4xy^2z + 4x^2yt + 2y^3t + 4x^2 - 10y^2 + 4xz - 10yt + 2, \\ 2yzt + xt^2 - x - 2z, \\ -xz^3 + 4yz^2t + 4xzt^2 + 2yt^3 + 4xz + 4z^2 - 10yt - 10t^2 + 2 \end{cases}$$

7.4. Modifyed cyclic 5-th roots

The next example shows that some systems are not stable at all. For instance if, in the cyclic 5-th roots system, we change the monomial $abcd$ into abc, we obtain a system which has nearly the same number of solutions, that is to say 64 instead of 70, but its Gröbner basis for the lexicographical ordering needs two hundred pages of listing and contains numbers with two hundred digits! It is the system:

$$\text{Mod} \quad \begin{cases} a + b + c + d + e, \\ ab + bc + cd + ae + de, \\ abc + bcd + abe + ade + cde, \\ bcd + abce + abde + acde + bcde, \\ abcde - 1 \end{cases}$$

7.5. Results.

In the first table, the columns correspond to five examples defined above. The computations are done on the rational number field or on the integers modulo 1831; the values in the table, which correspond to this latter case, are denoted by $\tilde{x}$. The rows are labelled by the following values:

n is the number of variables.

D is the number of solutions of the system.

d_{inp} is the maximal degree of the input polynomials.

τ is the CPU time of the computation. With a subscript, it is the time of Buchberger's algorithm for computing the Gröbner basis for the corresponding ordering (degree reverse lexicographical or lexicographical); without index, it is the time for passing from the first ordering to the second with *NewBasis*. By convention, a value of ∞ means that the computation ran out of memory without giving any result.

G is the saving of time, i.e. $G = \frac{\tau_{lex}}{\tau_{deg}+\tau}$.

d_{out} is the degree of the univariate polynomial of the Gröbner basis for lexicographical ordering. This measure, in some way, "the regularity of a system", because for a random choice of inputs, we have $D = d_out$.

k is the number of polynomials which appear in the Gröbner basis for the ordering labelled in subscript.

h is the height of coefficients in the Gröbner basis (number of decimal digits).

The following values have been computed for comparing the complexity with D^3 and nD^3.

α is the constant which would appear in the complexity of *NewBasis* if it were $O(d^3)$, that is $\alpha = \frac{\tau \times 1000}{D^3}$ where τ is in seconds. The scale of 1000 has been chosen in order to get values which are close to 1.

β is an heuristic measure of the exponent of the complexity: We suppose that $\tau = c\,D^{\beta}$, and we compute c from this formula applied to example C(6) and $\beta = 3$ (because linear algebra has a generally a complexity of $O(n^3)$); thus $\beta = \frac{\log \tau - \log c}{\log D}$.

β' : The same as β, but we suppose that $\tau = c\,n\,D^{\beta'}$; we also adjust the constant with system C(6) and $\beta' = 3$; thus $\beta' = \frac{\log \tau - \log(c\,n)}{\log N}$

Table 1.

	C(5)	C(6)	Aux	Cap	Mod
n	5	6	3	4	5
D	70	156	20	56	64
d_{inp}	5	6	6	4	5
d_{out}	15	48	20	20	61
k_{lex}	10	17	3	7	16
k_{deg}	19	44	15	24	23
h_{deg}	3	11	4	3	9
h_{lex}	6	23	9	8	207
τ_{lex}	24'29"	∞	1h25'52"	31'38"	∞
τ_{deg}	93"	3h24'50"†	1'31"	33"	26'42"
τ	1'55"	33'11"	56"	1'27"	2h37'15"
$\widetilde{\tau}_{lex}$	14'8"	∞	8'30"	9'44"	∞
$\widetilde{\tau}_{deg}$	59"	45'54"	47"	32"	14'56"
$\widetilde{\tau}$	1'56"	29'41"	31"	1'28"	28'2"
G	7	∞	35	15.8	∞
$\widetilde{G}$	4.9	∞	6.5	4.9	∞
α	0.335	0.52	35	0.5	36
β	2.89	3	3.86	2.99	4.02
β'	2.94	3	4.10	3.16	4.06
$\widetilde{\alpha}$	0.34	0.47	3.875	0.5	6.42
$\widetilde{\beta}$	2.91	2.99	3.68	3.0	3.61
$\widetilde{\beta'}$	2.79	2.85	3.68	3.0	3.5

† This computation has been done with a Lisp program. With the algorithm of SCRATCHPAD II it would take nearly 82 hours of CPU.

It should be noticed that the size of the coefficients (parameter h) is always much bigger for the lexicographical ordering than for the degree ordering; thus the difficulty of computing Gröbner basis for the lexicographical ordering appears also in the size of the output.

Values of α and β previously given show that for 'reasonable systems' ('reasonable' means that size of results are not too big) a good approximation of complexity is $O(D^3)$.

Next table corresponds to the time which is actually needed for the same examples (on a Sun station Sparc 10). As it appears, our algorithm is slower than the direct computation of the Gröbner base for the lexicographical ordering, when the latter is easy; nevertheless it remains useful for more difficult examples, and allows to compute Gröbner bases which would remain impossible without it.

Empty entries correspond to computations which have not been tried, being already too long with modular integers.

Table 2.

	C(5)	C(6)	C(7)	Aux	Cap	Mod
n	5	6	7	3	4	5
D	70	156	924	20	56	64
τ_{lex}	0"52	>7h41'		46"43	0"23	
τ_{deg}	0"35	13"82	24h26'40"†	0"17	0"20	4"67
τ	0"73	22"17	30h23'45"	1"55	0"75	16'35"
$\widetilde{\tau}_{lex}$	0"18	9"85	>5h	0"15	0"12	>5h‡
$\widetilde{\tau}_{deg}$	0"12	1"97	8'20"	0"05	0"05	0"78
$\widetilde{\tau}$	1"10	11"30	38'4"	0"13	0"60	5"

8. References.

Auzinger, W., Stetter, J. (1988). An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations, *Int. Ser. Num. Math.* **86**, 11–30.

Bareiss, E.H. (1968). Sylvester's Identity and Multistep Integer–preserving Gaussian Elimination. *Math. Comp.*, **22**, 565–578.

Becker, T., Weispfenning, V., (1993). *Gröbner bases* (in cooperation with H. Kredel). Springer.

Buchberger, B. (1965). Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. *Ph. D. Thesis,* Innsbruck.

Buchberger, B. (1970). Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleischunssystem, *Aeq. Math.* **4**, 374–383.

Buchberger, B. (1979). A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Basis. *Proc. EUROSAM 79, Lect. Notes in Comp. Sci.* **72**, Springer Verlag, 3–21.

Buchberger, B. (1985). Gröbner Bases : an Algorithmic Method in Polynomial Ideal Theory. In *Recent trends in multidimensional system theory,* Bose Ed., Reidel.

† With a new, not yet published algorithm.

‡ 155 Mbyte of data when stopped.

Caniglia, L., Galligo A. and Heintz J. (1988). Some new effectivity bounds in computational geometry. *Proceedings of AAECC-6,* Lect. Notes in Comp. Sci. **357**, Springer Verlag, 131–152.

Caniglia, L., Galligo A. and Heintz J. (1991). Equations for the projective closure and effective Nullstellensatz. *Discrete Applied Math.*, **33**, 11–23.

Davenport, J.H. (1987). Looking at a set of equations. Technical report 87–06, University of Bath.

Davenport, J.H., Siret, Y. and Tournier, E. (1986). *Calcul Formel, Systèmes et Algorithmes de Manipulation Algébriques.* Masson. English translation: *Computer Algebra.* Academic Press, 1988.

Gianni, P., Mora, T., (1987). Algebraic Solution of Systems of Polynomial Equations Using Gröbner Bases. Applied Algebra, Algebraic Algorithms and Error Correcting–Codes (AAECC–5), *Lect. Notes in Comp. Sci.*, **356**, 247–257.

Gianni, P., Trager, B., Zacharias, G. (1988). Gröbner Bases and Primary Decomposition of Polynomial Ideals. In *J. Symbolic Computation* **6**, 149–167.

Giovini, A., Mora, T., Niesi, G., Robbiano, L., Traverso, C. (1991) "One sugar cube, please"; or: Selection strategies in Buchberger algorithm. *Proc. ISSAC '91*, ACM, 49–54

Kobayashi, H., Moritsugu, S. and Hogan, R.W. (1988). On Solving Systems of Algebraic Equations. Symbolic and Algebraic Computation, International Symposium ISSAC '88 (P. Gianni, ed.). *Lect. Notes in Comp. Sci.* **358**, 139–149.

Lazard, D. (1983). Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations. *Proc. EUROCAL 83.* Lect. Notes in Comp. Sci. **162**, 146–157.

Lazard, D. (1989). Solving zero–dimensional algebraic systems. *J. of Symb. Comp.*, **13**, 117–132.

Möller, H. M. (1993). Systems of algebraic equations solved by means of endomorphisms, *Proceedings of AAECC-10*, Lect. Notes in Comp. Sci. **673** 43–56.

Trinks, W. (1978). Über B. Buchbergers Verfahren, Systeme Algebraischer Gleichungen zu Lösen. *J. Number Th.* **10**, 475–488.