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

8 References.

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.

Caniglia, L., Galligo A. and Heintz J. (1989a). Some new effectivity bounds in computational geometry. *Proceedings of AAECC-6, Roma 1988, Lect. Notes in Comp. Sci.***357**, Springer Verlag, 131–152.

Caniglia, L., Galligo A. and Heintz J. (1989b). How to compute the projective closure of an affine algebraic variety in subexponential time. To appear in *Proceedings of AAECC-7, Toulouse, 1989*

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.

Kobayashi, H., Moritsugu, S. and Hogan, R.W. (1988). On Solving Systems of Algebraic Equations. *Proc. ISSAC 88.*

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

Lazard, D. (1989). Solving zero–dimensional algebraic systems. *To appear.*

	(I)	(II)	(III)	(IV)	(V)
n	5	6	3	4	5
D	70	156	20	56	64
d_{inp}	5	6	6	4	5
d	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}	1469 sec	∞	5152 sec	1898 sec	∞
$\widetilde{\tau_{lex}}$	848 sec	∞	510 sec	584 sec	∞
τ_{deg}	93 sec	12290 sec[1]	91 sec	33 sec	1602 sec
$\widetilde{\tau_{deg}}$	59 sec	2754 sec	47 sec	32 sec	896 sec
τ	115 sec	1991 sec	56 sec	87 sec	9435 sec
$\tilde{\tau}$	116 sec	1781 sec	31 sec	88 sec	1682 sec
$\widetilde{G}$	4.9	∞	6.5	4.9	∞
G	7	∞	35	15.8	∞
α	0.335	0.52	35	0.5	36
$\tilde{\alpha}$	0.34	0.47	3.875	0.5	6.42
β	2.89	3	3.86	2.99	4.02
$\tilde{\beta}$	2.91	2.99	3.68	3.0	3.61
β'	2.94	3	4.10	3.16	4.06
$\widetilde{\beta'}$	2.79	2.85	3.68	3.0	3.5

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 bases 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 $O(D^3)$.

[1]About 3h36min. This computation has been done with a Lisp program. With the algorithm of SCRATCHPAD II it would take nearly 82 hours of CPU.

- τ is the time. With a subscript, it is the time of Buchberger's algorithm for computing the Gröbner base 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 *Newbase*. By convention, a value of ∞ means that the computation ran out of memory without giving any result.

- G is the saving of time that is $G = \frac{\tau_{lex}}{\tau_{deg}+\tau}$.

- d is the degree of the univariate polynomial of the Gröbner base for lexicographical ordering. This measure, in some way, " the regularity of a system", because for a random choice of inputs, we have $D = d$.

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

- h is the height of coefficients in the Gröbner base (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 *Newbase* if it were $O(d^3)$, that is $\alpha = \frac{\tau \times 1000}{D^3}$ where τ is in seconds.

- β is a measure of the exponent of the complexity : We suppose that $\tau = c\, D^{\beta}$, and we adjust c by choosing $\beta = 3$ in example (II) ; 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 (II) and $\beta' = 3$; thus

$$\beta' = \frac{\log \tau - \log(C\, n)}{\log N}$$

7.3 The Caprasse system

This is the system:

$$
(\mathrm{IV}) \begin{cases} y^2\ z + 2\ x\ y\ t - 2\ x - z = 0 \\ -x^3\ z + 4\ x\ y^2\ z + 4\ x^2\ y\ t + 2\ y^3\ t + 4\ x^2 \\ \qquad -10\ y^2 + 4\ x\ z - 10\ y\ t + 2 = 0 \\ 2\ y\ z\ t + x\ t^2 - x - 2\ z = 0 \\ -x\ z^3 + 4\ y\ z^2\ t + 4\ x\ z\ t^2 + 2\ y\ t^3 + 4\ x\ z \\ \qquad +4\ z^2 - 10\ y\ t - 10\ t^2 + 2 = 0 \end{cases}
$$

7.4 Variant of Arnborg with five variables.

The next example shows that some systems are not stable at all. For instance if, in Arnborg system with five variables, we change the $abcd$ monomial 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 base for the lexicographical ordering needs two hundred pages of listing and contains numbers with two hundred digits !
This is the system:

$$
(V) \begin{cases} a + b + c + d + e = 0 \\ a\ b + b\ c + c\ d + a\ e + d\ e = 0 \\ a\ b\ c + b\ c\ d + a\ b\ e + a\ d\ e + c\ d\ e = 0 \\ b\ c\ d + a\ b\ c\ e + a\ b\ d\ e + a\ c\ d\ e + b\ c\ d\ e = 0 \\ a\ b\ c\ d\ e - 1 = 0 \end{cases}
$$

7.5 Results.

In the following table, the columns correspond to the five above examples. The computations are done on the rational number field or on the integers modulo 1831 ; the values in the table, corresponding to the 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.

computation than for a direct computation of second Gröbner base (polynomial in d^{n^2}, Caniglia et al., 1989a and 1989b). Thus we can not enforce the experimental comparisons by a complexity result.

7 Examples.

We define now some very typical algebraic systems, then we shall sum up in a list various computer times achieved by a computer IBM 4381 (3.5 to $4Mips$).

7.1 Arnborg system with n variables.

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.

We define the Arnborg system with n variables by:

$$\begin{cases} x_1 + x_2 + \ldots + x_n = 0 \\ \sum_{\sigma \in G_n} x_{\sigma(1)} x_{\sigma(2)} = 0 \\ \sum_{\sigma \in G_n} x_{\sigma(1)} x_{\sigma(2)} x_{\sigma(3)} = 0 \\ \cdots\cdots\cdots \\ \sum_{\sigma \in G_n} x_{\sigma(1)} \ldots x_{\sigma(n-1)} = 0 \\ x_1 \ldots x_n = 1 \end{cases}$$

Examples (I) and (II) are this system with $n = 5$ or 6 respectively.

7.2 Modified Arnborg system with 7 variables.

This is the Arnborg system with seven variables 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:

$$\text{(III)} \begin{cases} a^2\ b\ c + a\ b^2\ c + a\ b\ c^2 + a\ b\ c + a\ b + a\ c + b\ c = 0 \\ a^2\ b^2\ c + a\ b^2\ c^2 + a^2\ b\ c + a\ b\ c + b\ c + a + c = 0 \\ a^2\ b^2\ c^2 + a^2\ b^2\ c + a\ b^2\ c + a\ b\ c + a\ c + c + 1 = 0 \end{cases}$$

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 of at most $k!$ products of k terms ; it is easy to verify that this produces results of polynomial size, proving the proposition.

Theorem 2 — *Newbase, with the first optimisation described in section* 4 *is polynomial in E over polynomial fields.*

Remark 5 — *We leave to the reader the determination of the degree of the complexity of Newbase 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 on 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 base.

Theorem 3 — *Let I be a zero–dimensional ideal of n–variate polynomials over a polynomial field, generated by polynomials of degree at most d ; suppose that I is also zero–dimensional at infinity, i.e. the homogeneous parts of highest degree of the generators have only a finite number of common zeros up to proportionality. Then, the Gröbner base of I for any ordering may be computed in a time polynomial in d^n.*

In Lazard, 1983, it is proven that the Gröbner base for the degree–reverse–lexicographical ordering may be computed in this time and contains polynomials of degree at most $nd-n+1$; it follows immediately from the proof that the monomials counted by E are at most of this degree and that $E \leq e^n d^n$ where e is the base of naturals logarithms. Thus applying *Newbase* on this base needs also a time which is polynomial in d^n.

Remark 6 — *If the ideal is zero–dimensional but not at infinity, all works well, but we have not a better complexity bound for the first Gröbner base*

In the computation of a normal form, a step of reduction of a polynomial p by an element f of the Gröbner base 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 base as entries.

Thus, the part of *Newbase* 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 4 — *A field is* polynomial*, if there exists polynomials 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 4 — *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.

the standard reduction process only on the product of a polynom in normal form by a variable.

On the other hand, we have just seen that D is not a good measure, that is a measure which leads to a polynomial algorithm.

Thus, we introduce a new measure, E which is the number of monomials which 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 *Newbase*.

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

Proposition 3 — *For a degree ordering, we have*

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

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

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

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 4 — *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 *Newbase* ; taking in 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 *Newbase* ; 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.

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 *Newbase*, 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.

Remark 3 — *If we try to take in account the growth of the coefficients, it appears that it is Matphi which introduces a growth which is exponential in nD ; in the next section, we will see that Newbase may be computed in a time which is exponential in n, but polynomial in D. We conjecture that this exponential behaviour is unavoidable.*

6 Complexity with growth of coefficients.

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, *Newbase* is mainly useful for changing from a degree ordering to another ordering.

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

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

As a polynomial in normal form with respect to *oldbase* 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 *oldbase* : the list of the irreducible monomials has size nD and each polynomial in the base is represented by its leading monomial (size n) and its reductum (size D).

By corollary 1, k is at most nD ; it is at least n for a zero–dimensional ideal ; thus :

Proposition 2 — *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 Newbase satisfies*

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

Remark 2 — *In practice, unless when D is very low, the number of elements of a Gröbner base is less than D, and $S \leq D^2 + 2nD$.*

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

Theorem 1 — *If the base field operations need an unit time, algorithm NewBase 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 Newbase is polynomial, on a field with unit cost for operations.

InsertNexts(monom) ;
for each j such that $monom = x_j m$
do

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

$monom := NextMonom$

end.

Before proving this algorithm, let us remark that the logical structure of it is exactly the same as in *Newbase* ; thus, the first test (divisibility by some leading term) does not need searching : As in *Newbase* it suffices to count the number of insertions in *ListOfNexts* for testing and to remember from which monomials *monom* was inserted for getting the decompositions $monom = x_i m$, with irreducible or reducible m . The second test (being the leading monomial) also does not need searching : If *oldbase* 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.

The proof of *Matphi* has been sketched in comments. In fact, this algorithm considers three cases for the monomials to be considered : the irreducible ones, those which appear in the leading term of some element of the Gröbner base, 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.

5 Complexity ; case of fields with unit cost.

We are now ready to compute the complexity of *NewBase*. The results are rather different if the operations in the field of coefficients take a constant time, or if we take in 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 representation, we need a convenient measure of the size of the input. It appears that two parameters are important :

- the number n of variables ;

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 decreasing order for $<$ and remove duplicates

Begin

monom := 1 *;*

ListOfNexts := [ ] *;*

while monom $\neq$ *nil do*

if monom is a strict multiple of the leading term of some element of *oldbase*

then

let *monom* $= x_j m$ with m reducible by *oldbase ;*

$[\![m < monom$; thus $NormalForm[monom] = \sum \lambda_i m_i$ has been previously computed with m_i irreducible, and $m_i < m]\!]$

$NormalForm[monom] := \sum \lambda_i NormalForm[x_j m_i]$ *;*

$[\![x_j m_i < m$, thus $NormalForm[x_j m_i]$ has been previously computed $]\!]$

for each j such that $monom = x_j m''$ with m irreducible by *oldbase*

do

$\phi[j,\ m',\ m'']$:= coefficient of m' in $NormalForm[monom]$

else if monom is the leading term of some element p of *oldbase*

then

$NormalForm[monom] := -rest(p)$ *;*

for each j such that $monom = x_j m$

do

$\phi[j,\ m',\ m]$:= coefficient of m' in $NormalForm[monom]$

else

$NormalForm[monom] := monom$ *;*

computed and stored in *vectbase*. 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 division 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*).

This method for computing the normal forms seems to be the best in practice; however, for computing the complexity of *Newbase* we need also another method, which we describe now.

This method is based on the fact that the application

$$\phi_i : m \rightarrow (x_i m)$$

is linear on polynomials in normal form ; if its matrix has been previously computed, the normal form computation of $x_i m$ is simply the product of a vector by this matrix.

We show now how to compute matrices ϕ_i in an efficient way.

Procedure 2 Matphi

Input :

> $<$*: an admissible ordering.*
>
> *oldbase,* a minimal reduced Gröbner base for a zero–dimensional ideal (as in *Newbase)*

Output :

> $\phi[i,\ m,\ m']$ *for* $i = 1$ *to* n*,* m *and* m' *in normal form* such that $\phi[i,\ *,\ *]$ is the matrix of the application
>
> $p \rightarrow NormalForm(x_i p)$ for p in normal form

Subfunctions

in *monbase* and maximal for this property ; thus the monomial, if not in *monbase*, is multiple of some element which appeared in *ListOfNexts* and were not put in *monbase* ; 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 *newbase* is a linear combination of elements of *monbase* and the normal form of an element of ideal(*oldbase*) is zero (first assertion above).

This proves that *newbase* is a standard base for the new ordering ; it is minimal and reduced as, by construct, no monomial of it is a multiple of a leading monomial of it other than itself.

We have now to prove that the algorithm *Newbase* stops ; we are in the zero–dimensional case ; so the maximal number of linearly independant 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 base. Thus the number of iterations which increase *monbase* is finite. When *monbase* is complete *ListOfNexts* may no more increase, thus the number of remainding 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 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 *monbase* by any variable is necessary in *monbase*, but this is false for the other monomials (see Proposition 1) ; 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.

4 Optimisation of *NormalForm*.

In Algorithm *Newbase*, all computations are now well defined, except *NormalForm*. In fact, the standard division algorithm works, but it is not optimal and does not permits us to compute the complexity. Thus we will give two optimisations of the normal forms computations.

For this purpose, let us remark that *NormalForm* is called on monomials of the form $m = m'x_i$ where m' is in *monbase* and its normal form has been

$vector + \sum_{v \in monbase} \lambda_v second(v) = 0$

then

$pol := monom \ + \sum_{v \in monbase} \lambda_v first(v)$;

newbase := *cons*(*pol, newbase*) ;

staircase := *cons*(*monom, staircase*) ;

else

monbase := *cons*([*monom.vector*], *monbase*) ;

InsertNexts(*monom*) ;

monom := *NextMonom*

end Newbase.

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

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

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

Secondly, the elements of *newbase* are in ideal(*oldbase*) : 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.

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

Now, each monomial which is not in *monbase* nor in *staircase* is multiple of some element of *staircase*. This is clear if the monomial has appeared in *ListOfNexts* ; if not, the monomial is of the form $m_1 m_2$ where m_1 is

<*: an new admissible ordering*

oldbase, a Gröbner base of a zero–dimensional ideal, for some ordering.

Output :

newbase, the reduced Gröbner base of the ideal generated by *oldbase* for ordering $<$.

Subfunctions :

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

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 decreasing order for $<$ and remove duplicates.

Local variables:

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

monbase, 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 *newbase* and $b_i = NormalForm(a_i)$.

Begin

$monbase := [\,]$ *;* $staircase := [\,]$ *;* $newbase := [\,]$ *;*

$monom := 1$ *;*

ListOfNexts $:= [\,]$ *;*

while $monom \neq nil$ *do*

  if *monom* is not a multiple of some element of *staircase*
  ⟦see below for an efficient implementation of this test ⟧
  then

    $vector := NormalForm(monom)$ *;*
    if there exist a linear relation

Corollary 1 *Let k be the number of generators of a reduced Gröbner basis for a zero dimensional ideal I ; then $k \leq nD(I)$.*

Remark 1 *This bound is generally very coarse, but it is attained when the set of irreducibles monomials is $\{1,\ x, \ldots, x^{d-1}\}$ for some variable x and some d.*

In most practicable examples we have k and $B(G)$ lower than D(I), and often much lower.

Definition 3 *Let us consider I, $(G,<)$, $B(G)$ and $M(G)$ as in Proposition 1 and let us order the elements of $M(G)$ with respect to $<$. We define $T(G)$ as the $|M(G)| \times D(I)$ matrix whose elements are :*

$$T(G)_{i,j} = j\text{–th coordinate on } B(G) \text{ of the reduction by } G \text{ of the } i\text{–th element of } M(G)$$

In next sections, we will give algorithms for computing efficiently $T(G)$ and which use it for computing the Gröbner base for a new ordering from an old one.

3 The algorithm.

Let K be a field and $R = K[x_1, \ldots, x_n]$ the ring of polynomials in $x_1, \ldots, x_n$ with coefficients in K. We consider I, a zero–dimensional ideal of polynomials given by its Gröbner base for some ordering. Zero–dimensional means that the set of common zeros of the polynomials in the ideal 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 base with a power of x_i as a leading monomial.

The following algorithm computes the Gröbner base of I for another ordering. In practice, the first ordering (the old one) will be a degree ordering, and the second one (the new ordering) will be the lexicographical one, but algorithm *Newbase* works in any case.

Procedure 1 Newbase

Input :

We recall now some definitions and properties of Gröbner bases that will be used in what follows.

Given an ideal I we will denote by $(G,<)$ the reduced Gröbner basis with respect to an admissible ordering $<$. We will say that an element $f \in R$ is *reduced* by G if no element $g \in G$ has a leading term that divides the leading term of f ; a Gröbner basis is *reduced* if each of its elements is reduced by the others.

Definition 1 *Given a zero dimensional ideal I in R and $(G,<)$ a reduced Gröbner basis for I, we will call* natural basis *of the K–vector space R/G, the basis $B(G)$ whose elements are the reduced monomials with respect to G. We will denote by $D(I)$ the dimension of the K–vector space R/G.*

Definition 2 *Let $B(G)$ be the natural basis for R/G, let*

$$M(G) = \{x_i b | \; b \in B(G), \;\; 1 \le i \le n, \;\; x_i b \notin B(G)\}$$

be the bordering of G.

The following proposition characterizes the elements of M.

Proposition 1 *Let I be a zero dimensional ideal, $(G,<)$ be the reduced Gröbner basis with respect to an admissible ordering $<$, and $B(G)$ be the natural basis of R/G then for every element $m \in M(G)$ exactly one of the following conditions holds :*

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

(ii) $m = x_j m_j$ for some j and some $m_j \in M(G)$

Proof.

(i) This is an immediate consequence of the definitions of a reduced Gröbner basis and of $B(G)$.

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

base (depending on the ordering), together with a discussion on the meaning of solving an algebraic system.

The Gröbner base 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 base of a system of polynomial equations of degree d in n variables is polynomial in d^n if the solutions are finite in number, counting the solutions at infinity (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 recently be proved to be $O(d^{n^2})$ when the number of solutions is finite (Caniglia et al. 1989a and 1989b) ; however, the base given by this ordering is well suited for computing the solutions (Trinks, 1978 ; Lazard, 1989).

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

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

This algorithm is implemented in Scratchpad II. Examples show that, for obtaining the base for the lexicographical ordering, it is much faster to compute the base for the degree ordering and to use our algorithm than a direct computation by Buchberger's algorithm. In an example we get the base for the lexicographical ordering where previous methods fail, even on supercomputers.

For positive dimensional case, we have some ideas for a generalization of our algorithm, but they do not yet work for computing the lexicographical Gröbner base and we do not yet know their efficiency in practice.

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 and by $I = (f_1, \ldots, f_m)$ a zero dimensional ideal in R generated by $f_1, \ldots, f_m$.

Efficient computation of zero–dimensional Gröbner bases by change of ordering

J.C. Faugère* P. Gianni† D. Lazard* T. Mora‡

October 30, 1991

Abstract

An efficient algorithm is given for computing the Gröbner base of a zero–dimensional ideal for some ordering from the Gröbner base corresponding to another ordering. This algorithm is polynomial in some natural sense. For obtaining the base for a lexicographical ordering, it is much faster to compute the base for a degree ordering and to use this new algorithm than to apply directly Buchberger's algorithm.

1 Introduction.

For solving algebraic systems, the main tool appears to be the computation of a Gröbner base (also called standard base) ; we refer to Buchberger, 1965, 1970, 1979, 1985 and Davenport and al., 1986 for the basic facts on this notion. The fact that the solutions come easily from the Gröbner base for the lexicographical ordering appear in Trinks, 1978. That the solutions of an algebraic system may be computed from the Gröbner base for an other ordering on the monomials appear in Kobayashi and al., 1988 ; in Lazard, 1989 several algorithms are given for computing the solutions from a Gröbner

*LITP, GRECO de Calcul Formel (CNRS), Université P. et M. Curie, 4 place Jussieu, F-75252 Paris Cedex 05

†Departimento di Matematica, Universitá, via Buonarroti 2, I–56100 Pisa.

‡Dipartimento di Matematica, Universitá, via L.B. Alberti 4, I-16132 Genova