

Changing the ordering of Gröbner Bases with LLL: Case of Two Variables

Abdolali Basiri, Jean-Charles Faugère

▶ **To cite this version:**

Abdolali Basiri, Jean-Charles Faugère. Changing the ordering of Gröbner Bases with LLL: Case of Two Variables. [Research Report] RR-4746, INRIA. 2003. inria-00071841

HAL Id: inria-00071841
https://inria.hal.science/inria-00071841

Submitted on 23 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Changing the ordering of Gröbner Bases with LLL: Case of Two Variables

Abdolali Basiri — Jean-Charles Faugère

N° 4746

24 Février 2003

THÈME 2



ISSN 0249-6399 ISRN INRIA/RR--4746--FR+ENG

INRIA LORRAINE

Changing the ordering of Gröbner Bases with LLL: Case of Two Variables

Abdolali Basiri*, Jean-Charles Faugère*

Thème 2 — Génie logiciel
et calcul symbolique
Projet Spaces, www-spaces.lip6.fr

Rapport de recherche n° 4746 — 24 Février 2003 — 14 pages

Abstract: We present an algorithm for the transformation of a Gröbner basis of an ideal with respect to any given ordering into a Gröbner basis with respect to any other ordering. This algorithm is based on a modified version of the LLL algorithm. The worst case theoretical complexity of this algorithm is not better than the complexity of the FGLM algorithm; but can also give the theoretical complexity with some parameters depending on the size of the output. When the output is small then algorithm is more efficient. We also present a first implementation of the algorithm in Maple. This algorithm is restricted to the case of two variables but works also in positive dimension.

Key-words: Gröbner basis, LLL, Reduced Lattice basis, Complexity.

* Laboratoire d'Informatique de Paris 6 (CNRS/UMR 7606), 4 place Jussieu, 75252 Paris Cedex 05, France, {basiri,jcf}@calfor.lip6.fr

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Change d'ordre d'une base de Gröbner avec LLL: le cas de deux variables

Résumé : Nous présentons un algorithme transformant une base de Gröbner d'un idéal pour un ordre donné en une base de Gröbner pour un autre ordre. Cet algorithme est basé sur une version modifiée de l'algorithme LLL. La complexité théorique, dans le pire des cas, n'est pas meilleure que la complexité de l'algorithme FGLM mais on peut exprimer cette complexité en fonction de paramètres dépendant de la taille de la sortie. Ainsi lorsque les degrés de la base de Gröbner finale sont petits l'algorithme devient plus efficace. Nous présentons une première implémentation en Maple de l'algorithme. L'algorithme est restreint au cas de deux variables mais est valide aussi en dimension positive.

Mots-clés : Bases de Gröbner, LLL, Réduction de réseaux, Complexité.

1 Introduction

One of the main tools for solving algebraic systems is the computation of Gröbner bases [Buc65, Buc70, Buc79, Buc85]. Gröbner bases for any ordering can be computed in one step with general algorithms [Buc65, Fau99, Fau02]. But, in practice, it is well known that it is often much faster to compute: a Gröbner basis G_1 for an ordering $<_1$; then to transform G_1 into another basis for another ordering $<_2$. When the ideal is zero–dimensional, the FGLM algorithm [FGLM93] is such an algorithm. The number of arithmetic operations of this algorithm is $O(nN_b^3)$ where n is the number of variables and N_b the number of solutions (with multiplicity). Transforming a Gröbner basis from a Total degree to a pure lexicographical ordering is the "natural way" for solving polynomial systems because, from a complexity point of view, the best ordering is the degree–reverse–lexicographical (DRL) one but it is easier to obtain the solutions (either numerically or symbolically) from a Gröbner basis for the lexicographical ordering. More surprisingly, experiments have shown that the other way (from a lexicographical ordering to a total degree ordering) is also very useful in many applications. For instance, to compute a decomposition into prime ideals it is often a good strategy to compute a lexicographical Gröbner basis of the projection on two variables, compute a decomposition of this projection and then to return to a total degree ordering to split the whole ideal (since it is a strong motivation for our algorithm we have included a sketch of this algorithm in section 8).

On the other hand LLL[LLL82] is a well known algorithm for computing short vectors in a lattice; since a Gröbner basis can be seen as the smallest polynomials in an ideal wrt the divisibility of their heading terms it is natural to compare the two algorithms. A possible benefit of LLL is that the complexity of this algorithm is very sensitive to the output; so one can hope that when the result is small the LLL algorithm should be faster than FGLM.

The difficulty in LLL is that it is difficult to "follow" symbolically. A standard algorithm for implementing the arithmetic of Jacobian groups of the curves C_{ab} is LLL; but in [BEFG02], we have replaced LLL by FGLM so that we can compute the Gröbner basis "by hand" on an generic input (that is to say with symbolic parameters as coefficients); so we were able to establish explicit formulas.

The plan of the paper is as follows. The section 3 is devoted to presenting the modified LLL algorithm. The section 4 includes also the proof of the correctness of the algorithm. The theoretical complexity of the algorithm is in 5 and the practical behavior of an implementation of the algorithm in Maple can be found in 6. The necessary mathematical notations are reviewed in section 2. For the sake of completeness we have also included a description of a meta algorithm for decomposing ideals in Annex 8 but this part may be omitted.

The name of this algorithm is simply algorithm LLL. In the rest of this paper LLL stands for this modified version of then LLL [LLL82] algorithm.

2 Definitions

In this paper we will denote by K a field, by $K[X,Y]$ the ring of polynomials in two variables with coefficients in K and by $I = (f_1, \ldots, f_m)$ an ideal in $K[X,Y]$ generated by $f_1, \ldots, f_m$.

We recall now some definitions and properties of Gröbner bases and lattice 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 $f \in K[X,Y]$ 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. We use the notations of [BW93]

Definition 1 *Let $(G = (g_1, \ldots, g_m), <)$ be a reduced Gröbner basis for I, we denote by r_i, the degree of $HT(g_i)$ w.r.t Y. The indices $(1, \ldots, m)$ can be considered in such away that $r_i \leq r_{i+1}$. For any positive integer number D_Y which is not less than $\deg_Y(G)$, let*

$$B_{D_Y}(G) = \{Y^{j_i} g_i |\ 1 \leq i \leq m,\ \ 0 \leq j_i \leq r_{i+1} - r_i - 1\}$$

where $r_{m+1} := D_Y$. We denote by $M_{D_Y}(G)$, the $K[X]$ submodule of $K[X,Y]$ generated by $B_{D_Y}(G)$ which is called D_Y-th $K[X]$ module associated to ideal I w.r.t $<$. In this case $B_{D_Y}(G)$ is called D_Y-th basis of $K[X]$ module associated to ideal I w.r.t $<$.

Let $\tilde{b}_1, \ldots, \tilde{b}_l$ be the linearly independent vectors in $K[X]^{D_Y}$ over $K(X)$, where l and D_Y be positive integer and $l \leq D_Y$. The lattice $L \subset K[X]^{D_Y}$ of rank m spanned by $\tilde{b}_1, \ldots, \tilde{b}_l$ is defined as

$$L = \sum_{i=1}^{l} K[X]\tilde{b}_i = \{\sum_{i=1}^{l} \lambda_i \tilde{b}_i |\ \lambda_i \in K[X],\ \ 1 \leq i \leq l\}.$$

We correspond to lattice $L \subset K[X]^{D_Y}$, the $K[X]$ submodule $M(L)$ of $K[X,Y]$ which is defined as

$$M(L) = \{\sum_{j=1}^{D_Y} \tilde{v_j} Y^{j-1} |\ \ \tilde{v} = (\tilde{v_1}, \ldots, \tilde{v}_{D_Y}) \in L\}.$$

Hence any vector $\tilde{v} = (\tilde{v_1}, \ldots, \tilde{v}_{D_Y}) \in L$ corresponds to $v = \sum_{j=1}^{D_Y} \tilde{v_j} Y^{j-1} \in M(L)$. Denote by $\tilde{B} = (\tilde{b}_{i,j})$ the $l \times D_Y$ matrix where $\tilde{b}_{i,j}$ is the j-th coordinate of $\tilde{b}_i$, and by $B = (b_{i,j})$ the $l \times D_Y$ matrix where $b_{i,j} = \text{coeff}(b_i, Y^{j-1})Y^{j-1} = \tilde{b}_{i,j} Y^{j-1}$. We define the determinant $d(L)$ of L to be the maximum of the determinant of $l \times l$ sub-matrices of B w.r.t order $<$. The orthogonality defect $OD(\tilde{b}_1, \ldots, \tilde{b}_l)$ of the basis $\tilde{b}_1, \ldots, \tilde{b}_l$ for the lattice L w.r.t order $<$, is defined as

$$HT(b_1) \ldots HT(b_l) - HT(d(L)).$$

Definition 2 *The basis $\tilde{b}_1, \ldots, \tilde{b}_l$ is called reduced if*

$$OD(\tilde{b}_1, \ldots, \tilde{b}_l) = 0.$$

For $1 \leq i \leq l$ a i-th *successive minimum* of $M(L)$ w.r.t order $<$ is a minimum element m_i of $M(L)$, such that m_i do not belong to the $K[X]$ submodule of $M(L)$, generated by $m_1, \ldots, m_{i-1}$. m_i is independent of the choice of $m_1, \ldots, m_{i-1}$. See [Mah41].

Proposition 3 *Let $\tilde{b}_1, \ldots, \tilde{b}_l$ be a reduced basis for a lattice $L \subset K[X]^{D_Y}$ of rank $l \leq D_Y$, Ordered in such away that $b_i \leq b_j$ for $1 \leq i < j \leq l$. Then b_i is a i-th successive minimum of $M(L)$ w.r.t order $<$ for $1 \leq i \leq l$.*

Proof: See [Len85]. □

Proposition 4 *Let $\tilde{b}_1, \ldots, \tilde{b}_l$ be a basis for a lattice $L \subset K[X]^{D_Y}$ of rank $l \leq D_Y$. If the coordinates of the vectors $\tilde{b}_1, \ldots, \tilde{b}_l$ can be permuted in such a way that they satisfy*

- $b_i \leq b_j \quad$ for $1 \leq i < j \leq l$
- $b_{i,j} < b_{i,i} \geq b_{i,k} \quad$ for $1 \leq i < j \leq l,\ i < k \leq D_Y$

then the basis $\tilde{b}_1, \ldots, \tilde{b}_l$ is reduced.

Proof: See [Pau98]. □

Theorem 5 *Let $(G = (g_1, \ldots, g_m), <)$ be a reduced Gröbner basis for I, D_Y a positive integer which is not less than $\deg_Y(G)$ and I_{D_Y} be the set of all polynomials in I whose degree respect to Y is less than D_Y, then $M_{D_Y}(G) = I_{D_Y}$.*

Proof: It is trivial that $M_{D_Y}(G) \subseteq I_{D_Y}$. If $M_{D_Y}(G) \neq I_{D_Y}$, let h be the minimum polynomials(w.r.t $<$) in I_{D_Y} which do not belong to $M_{D_Y}(G)$. Let $HT(h) = X^s Y^r$ and for $1 \leq i \leq m$, $HT(g_i) = X^{s_i} Y^{r_i}$, we can consider $r_i \leq r_{i+1}$. Since G is reduced we have $r_i < r_{i+1}$ and $s_i > s_{i+1}$ for $1 \leq i \leq m-1$. (If $r_i = r_{i+1}$ for some $1 \leq i \leq m-1$ then $HT(g_i) | HT(g_{i+1})$ or $HT(g_{i+1}) | HT(g_i)$, and if $s_i \leq s_{i+1}$ then $HT(g_i) | HT(g_{i+1})$ because $r_i < r_{i+1}$.)Let

$$i_0 = \max\{i \leq m \mid HT(g_i) | HT(h)\}$$

we clime that $i_0 = m$ or $r_{i_0} \leq r < r_{i_0+1}$. Otherwise there exist $r_{i_0} < r_{i_0+1} \leq r$ then since $s_{i_0+1} < s_{i_0} \leq s$ we will have $HT(g_{i_0+1}) | HT(h)$ which is a contradiction with choice of i_0.

Thus we have $r - r_{i_0} \leq r_{i_0+1} - r_{i_0} + 1$ or $r - r_{i_0} = r - r_m \leq D_Y - r_m$ whence there is $b = Y^{r - r_{i_0}} g_{i_0}$ into $B_{D_Y}(G)$ such that $HT(b) = X^{s_{i_0}} Y^r$. Now if we put

$$\tilde{h} = h - \frac{HM(h)}{HM(b)} b = h - \frac{HC(h)}{HC(b)} X^{s - s_{i_0}} b$$

then we will have $\tilde{h}$ belongs to $M_{D_Y}(G)$ and $HT(\tilde{h}) < HT(h)$ thus $\tilde{h} < h$, on the other hand $\tilde{h}$ belongs to I_{D_Y}, which is a contradiction with choice of h. Hence $M_{D_Y}(G) = I_{D_Y}$. □

3 The algorithm.

In this section we present a new version of the LLL algorithm [LLL82] which compute a Gröbner basis for some ordering from the Gröbner base corresponding to another ordering in $K[X,Y]$.

Algorithm 6 (LLL-Paulus)
Input : $(G_{old} = (g_1, \dots, g_m), <_{old})$ a reduced Gröbner basis for I, $<_{new}$, and D_Y a positive integer sufficiently large
Output : $G_{new} = (a_1, \dots, a_l)$ a Gröbner basis for I w.r.t $<_{new}$

$(b_1, \dots, b_l) \leftarrow \text{ModuleBasis}(G_{old}, <_{old}, D_Y)$

$k \leftarrow 0$

while $k < l$ **do**

Choose $i_0 \in \{k+1, \dots, l\}$ s.t $b_{i_0} = \min_{<_{new}}\{b_i \ : \ k+1 \leq i \leq l\}$, $swap(\tilde{b}_{k+1}, \tilde{b}_{i_0})$
Choose $j \in \{1, \dots, D_Y\}$ s.t $HT_{new}(b_{k+1}) = HT_{new}(b_{k+1,j})$
if $j \leq k$ **then**
$c \leftarrow b_{k+1} - \frac{HC_{new}(b_{k+1})}{HC_{new}(a_j)} X^{\deg(\tilde{b}_{k+1,j}) - \deg(\tilde{a}_{j,j})} \cdot a_j$
$p \leftarrow \max\{0 \leq s \leq k \ : \ a_s \leq_{new} c\}$
for $i = k+1$ **down to** $p+2$ **do** $\tilde{b}_i \leftarrow \tilde{a}_{i-1}$
$\tilde{b}_{p+1} \leftarrow \tilde{c}$
$k \leftarrow p$
else
$\tilde{c} \leftarrow \tilde{b}_{k+1}$
$\tilde{a}_{k+1} \leftarrow \tilde{c}$
Permute $(k+1, \dots, n)$ s.t. $HT_{new}(a_{k+1,k+1}) = HT_{new}(a_{k+1})$
$k \leftarrow k+1$

The sub-algorithm ModuleBasis compute the D_Y-th $K[X]$ module associated to ideal I w.r.t $<_o ld$.

ModuleBasis 7
Input: $(G = (g_1, \dots, g_m), <)$ a reduced Gröbner basis for I and D_Y, a positive integer sufficiently large
Output: $B_{D_Y}(G) = (b_1, \dots, b_l)$

$l \leftarrow D_Y - r_1 + 1$

$k \leftarrow 0$

Permute the coordinates $(1,\ldots,m)$ s.t $r_i < r_{i+1}$ where $HT(g_i) = X^{s_i}Y^{r_i}$

for i **from** 1 **to** $m-1$ **do**

for j **from** 0 **to** $r_{i+1} - r_i - 1$ **do**

$k \leftarrow k+1$

$b_k \leftarrow Y^j g_i$

for j **from** 0 **to** $D_Y - r_m$ **do**

$k \leftarrow k+1$

$b_k \leftarrow Y^j g_m$

4 Correctness

Theorem 8 *The algorithm 6 computes a Gröbner basis G_{new} in $K[X,Y]$ such that $Id(G_{old}) = Id(G_{new})$.*

Proof: Let D_Y be a positive integer such that

$$D_Y = \max\{\deg_Y(G_{old}), \deg_Y(G)\}$$

where G is a Gröbner basis for I w.r.t. $<_{new}$. For example we can consider $D_Y = 2\max\deg(G_{old}) - 1$ if the old ordering is the lexicographical ordering and the new ordering is the degree–reverse–lexicographical ordering, see [Buc83, Laz83]. Also if the degree of the new ordering w.r.t Y is at most equal to the degree of the old ordering w.r.t Y then we can consider $D_Y = \deg_Y(G_{old})$.

Termination: The number of passages in the principal loop is finite because in the k-th step

$$HT(a_1)\ldots HT(a_k)HT(b_{k+1})\ldots HT(b_l)$$

becomes smaller when k becomes also smaller or k stays unchanged, and it stays unchanged when k is increased by 1, and algorithms terminates when $k = l$ or equivalently when

$$HT(a_1)\ldots HT(a_l) = d(L).$$

Hence the number of passages in the principal loop is finite, and algorithms terminates when $k = l$.

Correctness: It is clear that $B_{D_Y}(G)$ and $(a_1,\ldots,a_l)$ generate the same $K[X]$ submodule M of $K[X,Y]$. By Theorem 5, $M = I_{D_Y}$. On the other hand by Proposition 4, $(\tilde{a}_1,\ldots,\tilde{a}_l)$ is a reduced basis for the lattice L with basis $(\tilde{b}_1,\ldots,\tilde{b}_l)$, because the following invariants are valid in the k-th step

- $a_i \leq a_j$ for $1 \leq i < j \leq k$

- $a_k \leq b_j$ for $k < j \leq l$
- $a_{i,j} < a_{i,i} > a_{i,t}$ for $1 \leq j < i \leq k$ and $i < t \leq D_Y$

Hence by Proposition 4, a_i is a $i-$th successive minimum of M and $HT(a_i) < HT(a_{i+1})$. (otherwise $HT(a_i) = HT(a_{i+1})$ thus $a' = a_{i+1} - a_i \in M$ and $HT(a') < HT(a_{i+1})$ which implies a' is dependent with $a_1, \ldots, a_i$, so $a_{i+1} = a' + a_i$ is also dependent with $a_1, \ldots, a_i$ which is a contradiction with choice of a_{i+1}). Now let f be a polynomial in $I_{D_Y} = M$, then there are $\lambda_1, \ldots, \lambda_l \in K[X]$ such that

$$f = \sum_{j=1}^{l} \lambda_j a_j$$

but for $1 \leq i < j \leq l$, $HT(\lambda_i a_i) \neq HT(\lambda_j a_j)$ because otherwise there are t_i, t_j s.t.

$$X^{t_i} HT(a_i) = HT(\lambda_i a_i) = HT(\lambda_i a_i) = X^{t_j} HT(a_j),$$

but $HT(a_i) < HT(a_j)$ implies $t_i > t_j$ (if $t_i < t_j$ then $X^{t_i} HT(a_i) < X^{t_j} HT(a_j)$, and if $t_i = t_j$ then $HT(a_i) = HT(a_j)$)hence $a' = X^{t_i - t_j} a_i - a_j \in M$ and $HT(a') < HT(a_j)$ which implies a' is dependent with $a_1, \ldots, a_{j-1}$, so $a_j = a' - X^{t_i - t_j} a_i$ depends with $a_1, \ldots, a_{j-1}$ which is a contradiction with the choice of a_{i+1}. Finally there is a unique $1 \leq j \leq l$ such that $HT(f) = HT(\lambda_j a_j)$, so $HT(a_j) | HT(f)$ which shows that $(a_1, \ldots, a_l)$ is a Gröbner basis for I w.r.t $<_{new}$. □

Remark. Unlike FGLM, this algorithm is not limited to zero dimensional ideals.

5 Complexity

We are now ready to compute the complexity of the algorithm 6 in terms of arithmetical operations in K. By an arithmetical operation in K, we mean addition, subtraction, multiplication or division of two elements of K. We use the notation of the algorithm 6. Denote by

- $D_Y = \max\{\deg_Y(G_{old}), \deg_Y(G_{new})\}$
- $l = D_Y - r_1$ where $r_1 = \deg_Y HT(g_1)$
- $d_X = \max\{deg_X(g_i) \mid 1 \leq i \leq m\}$
- $D_X = \max\{deg_X(a_i) \mid 1 \leq i \leq l\}$

Theorem 9 *Algorithm 6 takes $O(D_Y^3 . D_X^2)$ arithmetical operations in K to compute a Gröbner basis G_{new} in $K[X,Y]$ such that $Id(G_{old}) = Id(G_{new})$.*

Proof: Every pass of the main loop consists of $O(l.D_X)$ operations in K Let us look at, in the worst case, the number of passages in the loop. In every pass of the principal loop, either k increased by 1 or $OD(\tilde{a}_1, \ldots, \tilde{a}_k, \tilde{b}_{k+1}, \ldots, \tilde{b}_l)$ decreases(w.r.t $<_{new}$), in the other word in this case we have

$$c = b_{k+1} - \frac{HC_{new}(b_{k+1})}{HC_{new}(a_j)} X^{\deg(b_{k+1,j}) - \deg(a_{j,j})} \cdot a_j,$$

and $HT(a_1) \ldots HT(a_k) HT(c) HT(b_{k+2}) \ldots HT(b_l)$, with respect to $<_{new}$, is less than $HT(a_1) \ldots HT(a_k) HT(b_{k+1}) HT(b_{k+2}) \ldots HT(b_l)$. Hence

$$\deg_X(HT(c)) \leq \deg_X(HT(b_{k+1})) + D_X$$

and

$$\deg_Y(HT(c)) \leq D_Y.$$

Note that for two extremes order, the degree reverse lexicographical ordering($<_{DRL}$), and the lexicographical ordering($<_{Lex}$) we have

- $X^{n_{1,0}} <_{Lex} \ldots <_{Lex} X^{n_{1,1}}$
- $X^{n_{2,0}} Y <_{Lex} \ldots <_{Lex} X^{n_{2,1}} Y$
- $X^{n_{3,0}} Y^2 <_{Lex} \ldots <_{Lex} X^{n_{3,1}} Y^2$

  $\vdots$
- $X^{n_{ld_Y,0}} Y^{ld_Y - 1} <_{Lex} \ldots <_{Lex} X^{n_{ld_Y,1}} Y^{ld_Y - 1}$
- $X^{n_{ld_Y+1,0}} Y^{ld_Y} <_{Lex} \ldots <_{Lex} X^{n_{ld_Y+1,1}} Y^{ld_Y}$

thus the number of passages in the loop, in this case, is

$$S = \sum_{i=1}^{ld_Y+1} n_{i,1} - n_{i,0}$$

but $\deg_X(HT(c)) \leq \deg_X(HT(b_{k+1})) + D_X$ implies that

$$n_{i,1} \leq n_{i+1,0} + D_X \quad for\ i = 1 \ldots ld_Y$$

and hence

$$S \leq n_{ld_Y+1,1} + ld_Y D_X - n_{1,0}$$

thus the number of passages in the loop, in this case, is bounded by $S = l.d_Y.D_X$.

In the other hand

- $X <_{DRL} Y$

  $\vdots$

- $X^{ld_X-1}Y^m <_{DRL} X^{ld_X-2}Y^{m+1} \ldots <_{DRL} X^{ld_X-D_Y-1}Y^{m+D_Y}$
- $X^{ld_X}Y^m <_{DRL} X^{ld_X-1}Y^{m+1} \ldots <_{DRL} X^{ld_X-D_Y}Y^{m+D_Y}$

thus the number of passages in the loop, in this case, is bounded by $S = l.d_X.D_Y$. Hence the number of passages in the loop, in the worst case, is bounded by

$$S = D_Y^2 D_X.$$

and the total complexity is $O(D_Y^3.D_X^2)$. □

Let I be a zero dimensional ideal and denote by N_b, the dimension of the K vector space $\frac{K[X,Y]}{I}$. In this case $D_X = N_b$ and $D_Y \leq \sqrt{N_b}$ thus algorithm 6 takes $O(N_b^{3.5})$ arithmetical operations in K to compute a Gröbner basis G_{new} in $K[X,Y]$ such that $Id(G_{old}) = Id(G_{new})$. The best case for this algorithm is when D_Y is small.

6 Experiments

We have implemented the algorithm 6 in *Maple V Release 5*. In tables 1 and 2 we give the timings for some well known examples (Pentium 3 at 800 Mhz). Some words of caution are necessary: the quality of the computer implementation of Gröbner bases computations may have a dramatic effect on their performance. On the other hand, a Maple implementation of such an algorithm is not an efficient implementation even if it is useful to test the correctness of the algorithm and to give *a rough idea* of its practical behavior. Of course this implementation can not be compared with a low level implementation in C (as in FGb for instance). Another consequence is that we must restrict ourselves to middle size benchmarks. For all this reasons we add also in tables 1 and 2 the number of arithmetical operations (namely the number of multiplications): this number does not depend on the implementation and so can give an estimate of the intrinsic practical complexity of the algorithm. We add also the values of the parameters l, d_X, D_X, D_Y and S for each examples.

From this table, a first conclusion is that: this not optimized implementation of LLL is always faster than the standard implementation of FGLM *in Maple*. It is also clear that the best case of the algorithm is when l is small for example: (examples rand50-17, rand100-34 and rand150-51 in the table) l becomes 3 and hence the complexity of the algorithm 6 is simply $O(l^3 N_b^2)$.

DRL → Lex	Nb ∗ in LLL	N_b	FGLM (sec)	LLL (sec)	l	d_X	D_X	D_Y	S
Cyclic5	45	55	1.7	0.9	9	12	15	8	29
cyclic6	50, 938	126	458.5	194.	13	24	48	12	1, 056
katsura7	1, 970, 0234	128	> 13, 000	6293.5	16	16	128	15	7, 687
fabrice24	21, 232	40	141.9	53.1	9	9	40	8	488
*dessin*1	42, 431	46	315.	71.	10	10	46	9	752
dessin2	27, 089	42	148.5	44.7	9	9	42	8	560
benchmarkD1	575	48	17.3	0.6	3	25	48	2	49
benchmarki1	177, 908	66	576.	424.9	12	11	66	11	1, 792
UteshevBikker	18, 267	36	43.	41.2	9	8	36	8	460
rand50	49, 647	50	413.2	125.2	10	10	50	9	807
rand100	756, 530	100	> 5000	2070.	14	14	100	13	4, 217
rand150	4, 051, 692	150	> 24, 000.	11, 937.	17	17	150	16	12, 054
rand50-17	1, 632	50	11.3	3.5	3	34	50	2	51
rand100-34	6, 666	100	153.8	21.4	3	67	100	2	102
rand150-51	14, 798	150	917.1	63.4	3	101	150	2	150

Table 1: Comparison FGLM/LLL (from DRL to Lex) modulo p.

Lex → DRL	Nb ∗ in LLL	N_b	FGLM (sec)	LLL (sec)	l	D_X	d_Y	D_Y	S
cyclic5	279	55	2.8	2.2	9	15	7	8	66
cyclic6	11, 070	126	82.5	51.6	13	48	6	12	707
katsura7	1, 941, 523	128	> 12, 000.	5, 776.	16	128	1	15	16, 138
fabrice24	56, 550	40	295.4	111.5	9	40	1	8	1, 524
dessin1	94, 325	46	622.6	191.9	10	46	1	9	2, 174
dessin2	62, 588	42	359.5	131.9	9	42	1	8	1, 608
benchmarkD1	1, 200	48	9.6	1.5	3	48	1	2	76
benchmarki1	285, 970	66	3, 353.	656.2	12	66	1	11	4, 572
UteshevBikker	46, 111	36	247.1	86.	9	36	1	8	1, 354
rand50	110, 774	50	361.02	235.9	10	50	1	9	2, 383
rand100	905, 447	100	> 5, 000.	2, 192.	14	100	1	13	9, 601
rand150	3, 052, 423	150	> 18, 000	8, 599.	17	150	1	16	21, 436
rand50-17	4, 896	50	16.6	6.6	3	50	1	2	115
rand100-34	19, 760	100	235.6	37.2	3	100	1	2	230
rand150-51	44, 394	150	1617.6	112.2	3	150	1	2	346

Table 2: Comparison FGLM/LLL (from Lex to DRL) modulo p.

7 Conclusion

We have presented a new version of the LLL algorithm for computing Gröbner bases by changing the ordering. We give a proof and the theoretical complexity of the algorithm.

A first implementation in Maple is also presented and the first experimental results are encouraging. An open issue is to generalized this technique to more than two variables.

8 Appendix: meta algorithm for decomposing an ideal into primes

We give now the sketch of an algorithm to *speed up* the decomposition of an ideal given by a finite set of generators into a prime ideals. More precisely if I is an ideal the decomposition into primes is:

$$\sqrt{I} = P_1 \cap \cdots \cap P_k$$

then the output of the algorithm algorithm is $[G_1, \ldots, G_k]$ where G_i is the Gröbner basis of P_k for any ordering. The following algorithm is in fact a "meta algorithm": it calls a general decomposition algorithm `DEC` (see for instance [BW93]) but the idea is to apply this general algorithm `DEC` *after* decomposing as much as possible the initial ideal. Hence, in the worst case, this algorithm is not more efficient than `DEC` but in practice it is often much faster when the dimension is ≤ 1. In [Fau01] we have used a similar method to decompose the cyclic 9 problem (dimension 1 degree 6156).

Input

F a finite subset of $k[x_1, \ldots, x_n]$

Output

a decomposition into primes of the ideal generated by F.

1) Gröbner

G a Gröbner basis of F for a DRL ordering.

2) Elimination

Compute, by changing the ordering, G_e a Gröbner basis for a block ordering $[x_1, \ldots, x_{n-2}]$ $[x_{n-1}, x_n]$. This is an elimination ordering and so $G_e = G_1 \cup G_2$ where $G_2 \in k[x_{n-1}, x_n]$.

3) Lexico

Compute, by changing the ordering, G_{lex} a lexicographical Gröbner basis of G_2

4) Decompose (2 variables)

Use [Laz85] to compute a decomposition into prime components:

$$G_2 = P_1 \cap \cdots \cap P_k$$

(each P_i is a lexicographical Gröbner basis). The main tool to obtain this decomposition is to compute gcd of polynomials (see [Laz85]) so this can be done efficiently.

5) DRL in two variables

For each prime components compute, by change of ordering, a Gröbner for a DRL ordering:

$$P'_i = \text{Gröbner}(P_i, DRL)\ i = 1, \ldots, k$$

6) DRL

$$G'_i = \text{Gröbner}(G_e \cup P'_i, DRL)\ i = 1, \ldots, k$$

7) Decompose (general)

$$\texttt{DEC}(G'_1) \cup \cdot \texttt{DEC}(G'_k)$$

For all G'_i compute a decomposition into primes of G'_i.

Key points for the efficiency of the algorithm are steps 1, 3 and 5. Steps 3 and 5 can be done with the algorithm LLL presented in this paper.

Acknowledgements: We would like to thank Guillaume Hanrot for our motivation for studying *LLL* and its application in Gröbner bases. We thank Andreas Enge and Nicolas Gurel for interesting discussions on *LLL*.

References

[BEFG02] Abdolali Basiri, Andreas Enge, Jean-Charles Faugère, and Nicolas Gürel. Fast arithmetics for superelliptic cubics. Extended version, available from http://-www.lix.polytechnique.fr/Labo/Andreas.Enge/vorabdrucke/super.ps, 2002.

[Buc65] Buchberger B. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, Innsbruck, 1965.

[Buc70] Buchberger B. An Algorithmical Criterion for the Solvability of Algebraic Systems. *Aequationes Mathematicae*, 4(3):374–383, 1970. (German).

[Buc79] Buchberger B. A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Basis. In *Proc. EUROSAM 79*, volume 72 of *Lect. Notes in Comp. Sci.*, pages 3–21. Springer Verlag, 1979.

[Buc83] B. Buchberger. A note on the complexity of constructing Gröbner bases. In *van Hulzen J.A. (ed.) EUROCAL '83, European Computer Algebra Conference*, volume 162 of *Lecture Notes in Computer Science*, pages 137–145. Springer, 1983.

[Buc85] Buchberger B. Gröbner Bases : an Algorithmic Method in Polynomial Ideal Theory. In Reidel, editor, *Recent trends in multidimensional system theory.* Bose, 1985.

[BW93] T. Becker and V. Weispfenning. *Gröbner bases*. Springer-Verlag, NewYork-Berlin-Heidelberg, 1993.

[Fau99] Faugère J.C. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, June 1999.

[Fau01] Faugère J.C. How my computer find all the solutions of cyclic 9. In *Lecture Notes Series on Computing*, volume 9. World Scientific Publishing Co., 2001.

[Fau02] Faugère J.C. A new efficient algorithm for computing Gröbner bases without reduction to zero F5. In T. Mora, editor, *Proceedings of ISSAC*. ACM Press, July 2002.

[FGLM93] Jean-Charles Faugère, Patrizia Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16:329–344, 1993.

[Laz83] Daniel Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In *van Hulzen J.A. (ed.) EUROCAL '83, European Computer Algebra Conference*, volume 162 of *Lecture Notes in Computer Science*, pages 146–157. Springer, 1983.

[Laz85] D. Lazard. Ideal Bases and Primary Primary Decomposition:Case of Two Variables. *Journal of Symbolic Computation*, 1(3):261–270, September 1985.

[Len85] A. K. Lenstra. Factoring multivariate polynomials over finite fields. In *J. Computer and System Sciences*, volume 30, 1985.

[LLL82] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[Mah41] K. Mahler. An analogue of minkowski's geometry of numbers in a field of series. In *Annals of Math*, volume 42, 1941.

[Pau98] Sachar Paulus. Lattice basis reduction in function fields. In J. P. Buhler, editor, *Algorithmic Number Theory — ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*, pages 567–575, Berlin, 1998. Springer-Verlag.



Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

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