Fast Algorithm for Change of Ordering of Zero-dimensional Gröbner Bases with Sparse Multiplication Matrices*

Jean-Charles Faugère
INRIA Paris-Rocquencourt, SALSA Project
UPMC Univ Paris 06, UMR 7606, LIP6
CNRS, UMR 7606, LIP6
4 place Jussieu, 75005 Paris, France
Jean-Charles.Faugere@inria.fr

Chenqi Mou
LMIB, SMSS, Beihang University
Beijing 100191, PR China
and
INRIA Paris-Rocquencourt, SALSA Project
UPMC Univ Paris 06, UMR 7606, LIP6
CNRS, UMR 7606, LIP6
4 place Jussieu, 75005 Paris, France
Chenqi.Mou@lip6.fr

ABSTRACT

Let $I \subset \mathbb{K}[x_1,\ldots,x_n]$ be a 0-dimensional ideal of degree D where $\mathbb{K}$ is a field. It is well-known that obtaining efficient algorithms for change of ordering of Gröbner bases of I is crucial in polynomial system solving. Through the algorithm FGLM, this task is classically tackled by linear algebra operations in $\mathbb{K}[x_1,\ldots,x_n]/I$. With recent progress on Gröbner bases computations, this step turns out to be the bottleneck of the whole solving process.

Our contribution is an algorithm that takes advantage of the sparsity structure of multiplication matrices appearing during the change of ordering. This sparsity structure arises even when the input polynomial system defining I is dense. As a by-product, we obtain an implementation which is able to manipulate 0-dimensional ideals over a prime field of degree greater than 30000. It outperforms the Magma/Singular/FGb implementations of FGLM.

First, we investigate the particular but important shape position case. The obtained algorithm performs the change of ordering within a complexity $O(D(N_1 + n\log(D)))$, where N_1 is the number of nonzero entries of a multiplication matrix. This almost matches the complexity of computing the minimal polynomial of *one* multiplication matrix. Then, we address the general case and give corresponding complexity results. Our algorithm is dynamic in the sense that it selects automatically which strategy to use depending on the input. Its key ingredients are the Wiedemann algorithm to handle 1-dimensional linear recurrence (for the shape position case), and the Berlekamp–Massey–Sakata algorithm from Coding Theory to handle multi-dimensional linearly recurring sequences in the general case.

Categories and Subject Descriptors

I.1.2 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation—*Algorithms*; F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical algorithms and problems*

General Terms

Algorithms

Keywords

Gröbner bases, Change of ordering, Zero-dimensional ideals, Sparse matrix, FGLM algorithm, Wiedemann algorithm, BMS algorithm

1. INTRODUCTION

Gröbner basis is a major tool in computational ideal theory [5, 8, 3], in particular for polynomial system solving. It is well-known that the Gröbner basis of an ideal with respect to (w.r.t.) the lexicographical ordering (LEX) holds good algebraic structures, and hence is convenient to use for polynomial system solving. From the computational point of view, the common strategy to obtain such a Gröbner basis is to first compute a Gröbner basis w.r.t. the degree reverse lexicographical ordering (DRL), which is usually easier to compute, and then convert its ordering to LEX.

With recent progress on Gröbner basis computations [10, 11], the first step above has been greatly enhanced, leaving the second step, namely changing orderings of Gröbner bases, as the bottleneck of the whole solving process. Hence, currently, efficient algorithms to perform the change of ordering are of crucial significance in polynomial system solving. Furthermore, some practical problems can be modeled directly as a change of ordering [6, 15]. The purpose of this paper is precisely to provide a faster algorithm to perform the change of ordering of Gröbner bases of 0-dimensional ideals.

There already exist a few algorithms for the change of ordering of Gröbner bases, for example the FGLM algorithm [12] for 0-dimensional ideals and the Gröbner walk for generic cases [7]. The number of field operations needed by the FGLM algorithm is $O(nD^3)$, where n is the number of variables and D is degree of the given ideal $I \subset \mathbb{K}[x_1,\ldots,x_n]$. We would like to mention that other algorithms have been proposed to change the orderings of triangular sets [16, 9] or using the LLL algorithm [1] in the bivariate case. The connection between the change of ordering and linear algebra is done through the multiplication matrices T_i which represents the multiplication by x_i in the quotient ring $\mathbb{K}[x_1,\ldots,x_n]/I$ viewed as a vector space. According to our experiments (see table 2), these matrices are sparse, even when the input polynomial system is dense. The proposed algorithm takes advantage of this sparsity structure to obtain good complexity and performances.

First the particular but important case when the 0-dimensional ideal I is in shape position is studied. We consider the sequence

*This work is supported by the EXACTA grant of the French National Research Agency (ANR-09-BLAN-0371-01) and the National Science Foundation of China (NSFC 60911130369)

ISSAC'11, June 8–11, 2011, San Jose, California, USA.

$[\langle \boldsymbol{r}, T_1^i \boldsymbol{e}\rangle : i = 0,\ldots,2D-1]$, where $\boldsymbol{r}$ is a randomly chosen vector and $\boldsymbol{e} = (1,0,\ldots)^t$ is the canonical vector representing the term 1 in $\mathbb{K}[x_1,\ldots,x_n]/I$. It is easy to see that the minimal polynomial f_1 in $\mathbb{K}[x_1]$ of this linearly recurring sequence is indeed a polynomial in the Gröbner basis of I w.r.t LEX ($x_1 < \cdots < x_n$) when $\deg(f_1) = D$; moreover, it can be computed by applying the Berlekamp–Massey algorithm [20]. Furthermore, we show in section 3.1 how to recover efficiently the other polynomials in the Gröbner basis by solving structured (Hankel) linear systems. Hence, we are able to propose a complete method for the change of ordering to LEX for ideals in shape position. Its complexity is $O(D(N_1 + n\log(D)))$, where N_1 is the number of nonzero entries in T_1. When $n \ll D$ this almost matches the complexity of computing the minimal polynomial.

Next, for general ideals to which the method above may be no longer applicable, we generalize the linearly recurring sequence to a n-dimensional array $E : (s_1,\ldots,s_n) \longmapsto \langle \boldsymbol{r}, T_1^{s_1}\cdots T_n^{s_n}\boldsymbol{e}\rangle$. The minimal set of generating polynomials for the linearly recurring relation determined by E is essentially the Gröbner basis of the ideal defined by E, and this polynomial set can be obtained via the Berlekamp–Massey–Sakata (BMS for short hereafter) algorithm from Coding Theory [18, 19]. With some modifications of the BMS algorithm, we design a method to change the ordering in the general case. The algorithm is deterministic and the complexity with LEX as the target ordering is $O(nD^3)$ in the worst case and $O(nD(N + \hat{N}\bar{N}D))$ otherwise, where N is the maximal number of nonzero entries in matrices $T_1,\ldots,T_n$, while $\hat{N}$ and $\bar{N}$ are respectively the number of polynomials and the maximal term number of all polynomials in the resulting Gröbner basis.

Combining the two methods above, we propose a fast deterministic algorithm for the change of ordering for 0-dimensional ideals. This algorithm works for any term ordering, but we restrict the description to the case where LEX is the target ordering. It selects automatically which method to use depending on the input. The efficiency of the proposed algorithm has been verified by experiments. The current implementation outperforms the FGLM implementations in Magma/Singular/FGb. Take for example the Katsura12 instance over $\mathbb{F}_{65521}$, an ideal in shape position of degree 2^{12}, the change of ordering to LEX can be achieved in 26.3 seconds: this is 53.7 (resp 99.8) faster than the corresponding Magma (resp. Singular) function. As shown in table 2, 0-dimensional ideals over a prime field of degree greater than 30000 are now tractable.

The organization of this paper is as follows. Related algorithms used in this paper, together with some notations, are first reviewed in Section 2. Then Section 3 is devoted to our main algorithm. The complexity analysis of this algorithm is stated in Section 4 and experimental results are given in Section 5 respectively. The proof of one main theorem in the complexity analysis depends on properties of objects arising in the BMS algorithm applied in our context. The proof being technical, it is postponed in Section 6. This paper concludes with some remarks in Section 7.

2. FGLM AND BMS ALGORITHMS

2.1 FGLM

The FGLM algorithm is an efficient approach to convert the Gröbner basis of a 0-dimensional ideal w.r.t. a term ordering to another term ordering [12].

Let $\mathbb{K}$ be a field and $\mathbb{K}[x_1,\ldots,x_n]$ be the polynomial ring over $\mathbb{K}$. Suppose now the Gröbner basis G_1 of a 0-dimensional ideal I w.r.t. $<_1$ is known and one wants to compute its Gröbner basis G_2 w.r.t. $<_2$ with the FGLM algorithm. Let D be the degree of I and $B = [\varepsilon_1,\ldots,\varepsilon_D]$ be the canonical basis of $\mathbb{K}[x_1,\ldots,x_n]/\langle G_1\rangle$ ordered according to $<_1$.

The algorithm first computes $(D\times D)$-matrices T_i, called the *multiplication matrix by* x_i, to record the mapping ϕ_i on B:

$$\phi_i(b_j) = \text{NormalForm}(x_i b_j), \quad j = 1,\ldots,D$$

for $i = 1,\ldots,n$, where NormalForm() is the normal form w.r.t. G_1. The jth column of T_i is the coordinate vector of NormalForm$(x_i b_j)$ w.r.t. B. Thus from the basis B, one can construct all the matrices $T_1,\ldots,T_n$ accordingly. As can be seen here, all T_i and T_j commute.

Then terms in $\mathbb{K}[x_1,\ldots,x_n]$ are handled one by one, following the term ordering $<_2$. For a term $\boldsymbol{x}^{\boldsymbol{s}}$ with $\boldsymbol{s} = (s_1,\ldots,s_n)$, its coordinate vector $\boldsymbol{v}_{\boldsymbol{s}}$ w.r.t. B can be computed by

$$\boldsymbol{v}_{\boldsymbol{s}} = T_1^{s_1}\cdots T_n^{s_n}\boldsymbol{e}, \tag{1}$$

where $\boldsymbol{e} = (1,0,\ldots,0)^t$ is the coordinate vector of the term 1. Then a linear dependency like

$$\sum_{\boldsymbol{s}} c_{\boldsymbol{s}}\boldsymbol{v}_{\boldsymbol{s}} = 0 \tag{2}$$

will furnish an element in G_2:

$$f = \boldsymbol{x}^{\boldsymbol{l}} + \sum_{\boldsymbol{s}\neq\boldsymbol{l}} \frac{c_{\boldsymbol{s}}}{c_{\boldsymbol{l}}}\boldsymbol{x}^{\boldsymbol{s}}, \tag{3}$$

where $\boldsymbol{x}^{\boldsymbol{l}}$ is the leading term of f w.r.t. $<_2$ (denoted by $\text{lt}(f)$) [12]. The test of linear dependency can be realized by maintaining an echelon form of the matrix whose columns are coordinate vectors of previously computed terms w.r.t. B.

As for its complexity, the FGLM algorithm needs $O(nD^3)$ field operations to finish the change of ordering.

2.2 BMS

The BMS algorithm is one that can be used to find the minimal set w.r.t. a term ordering $<$ of a linearly recurring relation generated by a given multi-dimensional array [18, 19, 17]. It is a generalization of Berlekamp–Massey algorithm, which determines the minimal polynomial of a linearly recurring sequence.

As a vector $\boldsymbol{u} = (u_1,\ldots,u_n) \in \mathbb{Z}_{\geq 0}^n$ and a term $\boldsymbol{x}^{\boldsymbol{u}} = x_1^{u_1}\cdots x_n^{u_n} \in \mathbb{K}[x_1,\ldots,x_n]$ are 1–1 corresponding, usually we do not distinguish them. Besides the term ordering, we define the following partial ordering: for two terms $\boldsymbol{u} = (u_1,\ldots,u_n)$ and $\boldsymbol{v} = (v_1,\ldots,v_n)$, we say that $\boldsymbol{u} \prec \boldsymbol{v}$ if $u_i \leq v_i$ for $i = 1,\ldots,n$.

A mapping $E : \mathbb{Z}_{\geq 0}^n \longrightarrow \mathbb{K}$ is called a *n-dimensional array*. For a polynomial $f = \sum_{\boldsymbol{s}} f_{\boldsymbol{s}} x^{\boldsymbol{s}} \in \mathbb{K}[x_1,\ldots,x_n]$, a n-dimensional array E is said to satisfy the n-dimensional linearly recurring relation with characteristic polynomial f if

$$\sum_{\boldsymbol{s}} f_{\boldsymbol{s}} E_{\boldsymbol{s}+\boldsymbol{r}} = 0, \quad \forall \boldsymbol{r} \succ 0. \tag{4}$$

The set of all characteristic polynomials of n-dimensional linearly recurring relations for the array E forms an ideal, denoted by $I(E)$. And the minimal set of generating polynomials for $I(E)$, which the BMS algorithm computes, is actually the Gröbner basis of $I(E)$ w.r.t. $<$ [19, Lemma 5]. The canonical basis of $\mathbb{K}[x_1,\ldots,x_n]/I(E)$ is also called the *delta set* of E.

Instead of studying the infinite array E as a whole, the BMS algorithm deals with a truncated subarray of E up to some term $\boldsymbol{u}$ according to a given term ordering. A polynomial f with $\text{lt}(f) = \boldsymbol{s}$ is said to be *valid for E up to* $\boldsymbol{u}$ if either $\boldsymbol{u} \not\succ \boldsymbol{s}$ or $\sum_{\boldsymbol{t}} f_{\boldsymbol{t}} E_{\boldsymbol{t}+\boldsymbol{r}} = 0, \forall \boldsymbol{r}(0 \prec \boldsymbol{r} \leq \boldsymbol{u} - \boldsymbol{s})$. E may be omitted if no ambiguity occurs.

Similar to FGLM, the BMS algorithm also handles terms in $\mathbb{K}[x_1,\ldots,x_n]$ one by one according to $<$, so that the polynomial set it maintains is valid for E up to the new term. Let $F \subset \mathbb{K}[x_1,\ldots,x_n]$ be a set of polynomials whose elements are all valid up to some term $\boldsymbol{u}$. When the next term of $\boldsymbol{u}$ w.r.t. the term ordering, denoted by Next($\boldsymbol{u}$), is considered, the BMS algorithm will update F so

that all the new polynomials in it are valid up to Next($\boldsymbol{u}$). Meanwhile, another term determined by Next($\boldsymbol{u}$) is also tested to see whether it is a member of the delta set of E. Therefore, more and more terms will be verified as members of the delta set of E while terms are handled by the BMS algorithm. The set of verified terms in the delta set of E after the term $\boldsymbol{u}$ is called the *delta set up to* $\boldsymbol{u}$. After a certain number of terms are considered, this polynomial set F grows to a minimal set of polynomials generating the linearly recurring relation, namely a Gröbner basis of $I(E)$, and all members in the delta set of E are verified.

Due to limited space, only outlines of the above update procedure (which is also the main part) in the BMS algorithm are summarized here so that this paper is self-contained. One may refer to [17] for details. The polynomial set G below, called the *witness set*, is auxiliary and will not be returned with F in the end.

Algorithm 1: $(F^+, G^+) :=$ BMSUpdate$(F, G, \text{Next}(\boldsymbol{u}), E)$

Input:
F, a minimal polynomial set valid up to $\boldsymbol{u}$;
G, a witness set up to $\boldsymbol{u}$;
Next($\boldsymbol{u}$), a term;
E, a n-dimensional array up to Next($\boldsymbol{u}$).

Output:
F^+, a minimal polynomial set valid up to Next($\boldsymbol{u}$);
G^+, a witness set up to Next($\boldsymbol{u}$).

1. Test whether every polynomial in F is valid up to Next($\boldsymbol{u}$)
2. Update G^+ and compute the new delta set up to Next($\boldsymbol{u}$) accordingly
3. Construct new polynomials in F^+ such that they are valid up to Next($\boldsymbol{u}$)

Let k be the number of terms the BMS algorithm has handled before it stops at some term when F is the Gröbner basis and l be the number of polynomials it returns. Then the claimed complexity of the BMS algorithm is $O(lk^2)$ for graded term orderings [19].

3. MAIN ALGORITHM

3.1 Shape position

An ideal $I \subset \mathbb{K}[x_1, \ldots, x_n]$ is said to be *in shape position* if its Gröbner basis w.r.t. LEX ($x_1 < \cdots < x_n$) is of the form

$$[f_1(x_1), x_2 - f_2(x_1), \ldots, x_n - f_n(x_1)]. \tag{5}$$

For exact characterization of such ideals, one may refer to [2]. Ideals in shape position take a large proportion in all the consistent ideals. Thanks to their structures, we are able to design a specific and efficient method for the change of ordering with LEX as the target ordering .

Suppose the Gröbner basis of the deal I w.r.t. LEX is of form (5), $\deg(f_1) = D$ and $f_i = \sum_{k=0}^{D-1} c_{i,k} x_1^k$ $(i = 2, \ldots, n)$, where $c_{i,k}$ are unknown coefficients in $\mathbb{K}$. We consider now the linearly recurring sequence

$$[\langle \boldsymbol{r}, T_1^i \boldsymbol{e} \rangle : i = 0, \ldots, 2D-1], \tag{6}$$

where $\boldsymbol{r} \in \mathbb{K}^{(D\times 1)}$ is a randomly chosen vector, T_1 is the matrix constructed in the FGLM algorithm, and $\boldsymbol{e}$ is the vector representing 1 w.r.t. the canonical basis of $\mathbb{K}[x_1, \ldots, x_n]/I$. We first recall some basic facts about linearly recurring sequences:

DEFINITION 3.1. *Let $T = [t_0, t_1, t_2, \cdots]$ be a sequence of elements of $\mathbb{K}$ and d an integer. We define the following $d \times d$ Hankel matrix:*

$$H_d(T) = \begin{bmatrix} t_0 & t_1 & t_2 & \cdots & t_{d-1} \\ t_1 & t_2 & t_3 & \cdots & t_d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{d-1} & t_d & t_{d+1} & \cdots & t_{2d-2} \end{bmatrix}.$$

THEOREM 3.1. ([14]) *Let $T = [t_0, t_1, t_2, \cdots]$ be a linearly recurring sequence. Then, the minimal polynomial $M^{(T)}(x) = \sum_{i=0}^{d} m_i x^i$ of the sequence T is such that:*

(i) $d = \text{rank}(H_d(T)) = \text{rank}(H_i(T))$ *for all* $i > d$.

(ii) $\ker(H_{d+1}(T))$ *is a vector space of dimension* 1 *generated by* $(m_0, m_1, \ldots, m_d)^t$.

Moreover, since a bound on the size of the linearly recurring sequence is known (D is always a bound), the Berlekamp–Massey algorithm can compute the minimal polynomial $\tilde{f}_1$ of the sequence (6) (it is also possible to use the *deterministic variant* of the Wiedemenann algorithm [20] to compute directly f_1). Next we check whether $\deg(\tilde{f}_1) = D$, which implies that $\tilde{f}_1 = f_1$. If it holds, then computing the full Gröbner basis of I w.r.t. LEX reduces to determining all the unknown coefficients $c_{i,k}$.

For each $i = 2, \ldots, n$, from NormalForm$(x_i - \sum_{k=0}^{D-1} c_{i,k} x_1^k) = 0$ one can get $\boldsymbol{v}_i := T_i \boldsymbol{e} = \sum_{k=0}^{D-1} c_{i,k} \cdot T_1^k \boldsymbol{e}$. Then one can further construct D linear equations,

$$\langle \boldsymbol{r}, T_1^j \boldsymbol{v}_i \rangle = \sum_{k=0}^{D-1} c_{i,k} \cdot \langle \boldsymbol{r}, T_1^{k+j} \boldsymbol{e} \rangle, \quad j = 0, \ldots, D-1. \tag{7}$$

With $c_{i,k}$ considered as unknowns, the coefficient matrix H with entries $\langle \boldsymbol{r}, T_1^{k+j} \boldsymbol{e} \rangle$ is a Hankel one. From theorem 3.1 we know that H is invertible. Furthermore, the linear equation set (7) with the Hankel matrix H can be efficiently solved with complexity $O(D \log(D))$ if fast polynomial multiplication is used [4]. In the end, the solution of (7) will lead to the Gröbner basis we want to compute.

We explain now how the linear systems (7) can be generated for free. Note that for any $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{K}^{(D\times 1)}$ and $T \in \mathbb{K}^{(D\times D)}$, we have $\langle \boldsymbol{a}, T\boldsymbol{b} \rangle = \langle T^t \boldsymbol{a}, \boldsymbol{b} \rangle$, where T^t denotes the transpose of T. Thus

$$\langle \boldsymbol{r}, T_1^i \boldsymbol{e} \rangle = \langle (T_1^t)^i \boldsymbol{r}, \boldsymbol{e} \rangle, \quad \langle \boldsymbol{r}, T_1^j \boldsymbol{v}_i \rangle = \langle (T_1^t)^j \boldsymbol{r}, \boldsymbol{v}_i \rangle$$

in (6) and (7). Therefore, when computing the sequence (6), we can record $(T_1^t)^i \boldsymbol{r}$ $(i = 0, \ldots, 2D-1)$ and use them for construction of the linear equation set (7).

3.2 General case

Now we demonstrate the method for the change of ordering of a 0-dimensional Gröbner basis in the general case. In what follows, we always assume that $\mathbb{K}$ is field of characteristic 0 or a finite field of large cardinality. This is because otherwise "bad" random vectors would be frequently chosen so that BMS algorithm may not work.

Define a mapping $E : \mathbb{Z}_{\geq 0}^n \longrightarrow \mathbb{K}$ as

$$(s_1, \ldots, s_n) \longmapsto \langle \boldsymbol{r}, T_1^{s_1} \cdots T_n^{s_n} \boldsymbol{e} \rangle,$$

where $\boldsymbol{r} \in \mathbb{K}^{(D\times 1)}$ is a random vector. Combining (1) and (2), one can easily verify that the polynomial f in (3) is a characteristic polynomial for the n-dimensional linearly recurring relation defined by the array E. That is, f in (3) satisfies (4). This observation links FGLM and BMS algorithms: one can first construct the n-dimensional array E via matrices $T_1, \ldots, T_n$, and then compute the Gröbner basis of $I(E)$ with the BMS algorithm w.r.t. $<_2$.

This idea can actually be regarded as a generalization of the Wiedemann algorithm to the multivariate case, with the BMS algorithm to compute the minimal set of generating polynomials just as the role of Berlekamp–Massey algorithm. Unfortunately, similar to the Wiedemann algorithm, the strategy used here may also fail returning all the linear dependencies needed by the FGLM algorithm to form the Gröbner basis w.r.t. $<_2$. That is to say, the polynomial set returned by the BMS algorithm may only be a Gröbner basis of

$I(E)$ instead of I, where $I \subset I(E)$. However, one can easily check whether the set returned by the BMS algorithm is a Gröbner basis of I or not by testing the linear dependency in (2).

We remark that when the target ordering is LEX, computation of the first characteristic polynomial in the method above is essentially the same as that based on the Wiedemann algorithm described in section 3.1. This is true because for the LEX ordering ($x_1 < \cdots < x_n$), the terms are ordered as $[1, x_1, x_1^2, \ldots, x_2, x_1x_2, x_1^2x_2, \ldots]$, hence the first part of E is $E((p_1, 0, \ldots, 0)) = \langle \boldsymbol{r}, T_1^{p_1}\boldsymbol{e}\rangle$, and the BMS algorithm degenerates to the Berlekamp–Massey one now.

3.3 Algorithm description

Here the description of the main algorithm with the target ordering as LEX is given, together with some explanations and comments.

In algorithm 2 below, BerlekampMassey() is the Berlekamp–Massey algorithm, which takes a sequence over $\mathbb{K}$ as input and returns the minimal polynomial of this sequence [20]; Reduce(F) performs reduction on F so that every polynomial $f \in F$ is reduced w.r.t. $F \setminus \{f\}$; IsGB(F) returns true if F is the Gröbner basis of I w.r.t. LEX, and returns false otherwise; FGLM() is the FGLM algorithm.

Algorithm 2: Main algorithm

Input: G_1, the Gröbner basis of a 0-dimensional ideal $I \subset \mathbb{K}[x_1, \ldots, x_n]$ w.r.t. $<_1$
Output: the Gröbner basis of I w.r.t. LEX

1 Compute the canonical basis $[\varepsilon_1 = 1 <_1 \cdots <_1 \varepsilon_D]$ of $\mathbb{K}[x_1, \ldots, x_n]/\langle G_1\rangle$
2 $\boldsymbol{e} := (1, 0, \ldots, 0)^t \in \mathbb{K}^{(D\times 1)}$
3 Compute $T_1, \ldots, T_n$ the multiplication matrices
4 Choose $\boldsymbol{r}_0 = \boldsymbol{r} \in \mathbb{K}^{(D\times 1)}$ randomly
5 **for** $i := 1, \ldots, 2D-1$ **do** $\boldsymbol{r}_i := (T_1^t)\boldsymbol{r}_{i-1}$
6 Generate the sequence $s := [\langle \boldsymbol{r}_i, \boldsymbol{e}\rangle : i = 0, \ldots, 2D-1]$
7 $f_1 :=$ BerlekampMassey(s)
8 **if** $\deg(f_1) = D$ **then**
9 $\quad H := H_D(s)$ *// Construct the Hankel matrix*
10 $\quad$ **for** $i := 2, \ldots, n$ **do**
11 $\quad\quad \boldsymbol{b} := \big(\langle \boldsymbol{r}_j, T_i\boldsymbol{e}\rangle : j = 0, \ldots, D-1\big)^t$
12 $\quad\quad$ Compute $\boldsymbol{c} = (c_1, \ldots, c_D)^t := H^{-1}\boldsymbol{b}$
13 $\quad\quad f_i := \sum_{k=0}^{D-1} \boldsymbol{c}_{k+1} x_1^k$
14 $\quad$ **end**
15 $\quad$ **return** $[f_1, x_2 - f_2, \ldots, x_n - f_n]$
16 **else**
17 $\quad \boldsymbol{u} := \boldsymbol{0}$; $F := [1]$; $G := [\,]$; $E := [\,]$ *// General case*
18 $\quad$ **repeat**
19 $\quad\quad e := \langle \boldsymbol{r}, T_1^{\boldsymbol{u}_1} \cdots T_n^{\boldsymbol{u}_n}\boldsymbol{e}\rangle$
20 $\quad\quad E := E \cup [e]$
21 $\quad\quad F, G :=$ BMSUpdate($F, G, \boldsymbol{u}, E$)
22 $\quad\quad \boldsymbol{u} :=$ Next($\boldsymbol{u}$) w.r.t. LEX
23 $\quad\quad F :=$ Reduce(F)
24 $\quad$ **until** *Termination Criteria* ;
25 $\quad$ **if** *not* IsGB(F) **then**
26 $\quad\quad F :=$ FGLM($G_1, <_1$)
27 $\quad$ **end**
28 $\quad$ **return** F
29 **end**

With earlier computed values $T_1^{\boldsymbol{u}_1} \cdots T_n^{\boldsymbol{u}_n}\boldsymbol{e}$ recorded, the computation of e at line 19 can be simplified. Suppose for $\boldsymbol{v} = (v_1, \ldots, v_n)$, the vector $\tilde{\boldsymbol{e}} = T_1^{\boldsymbol{v}_1} \cdots T_i^{\boldsymbol{v}_i - 1} \cdots T_n^{\boldsymbol{v}_n}\boldsymbol{e}$ has been recorded. Then $\langle \boldsymbol{r}, T_1^{\boldsymbol{v}_1} \cdots T_n^{\boldsymbol{v}_n}\boldsymbol{e}\rangle = \langle \boldsymbol{r}, T_i\tilde{\boldsymbol{e}}\rangle$, for all T_i and T_j commute.

Though the BMS algorithm from Coding Theory is mainly designed for graded term orderings, it works for all term orderings. However, for orderings that depend on lexicographical orderings (for instance LEX or block orderings which break ties with LEX), some other techniques not mentioned in the original presentation of BMS algorithm should be used. For example, the reduction step is introduced to control the size of intermediate polynomials. This is actually not a problem for orderings like DRL, for in that case the leading term of a polynomial itself will give a bound on the size of terms in that polynomial.

Unfortunately the termination criteria of the BMS algorithm are not well studied in the literature. There exist some general (necessary or sufficient) conditions on when the polynomial set F the BMS algorithm maintains will eventually become the Gröbner basis [19], but basically they are not very suitable to use as termination criteria. Hence here we mainly use the criterion that the main loop (lines 18–24) ends when F keeps unchanged for a certain number of passes.

REMARK 3.1. To change algorithm 2 to one suitable for all target orderings, one only needs to skip lines 5–15, that is, the method designed for ideals in shape position for LEX.

3.4 Correctness and termination

Now we are in a position of proving the correctness and termination of algorithm 2 proposed above.

Correctness. Lines 5–15 are for ideals in shape position, the correctness of this method is obvious from its description in section 3.1. After the termination criteria are reached, the main loop ends and whether the returned polynomial set F is the Gröbner basis of I w.r.t. LEX is tested. If IsGB(F) = true, F is already the Gröbner basis we want to compute and the algorithm naturally finishes. While IsGB(F) = false means that the BMS algorithm returns a polynomial set F which is only Gröbner basis of $I(E)$, but $I(E) \neq I$ (on the assumption that the termination criteria work). Then we have to return to the original FGLM algorithm to complete the change of ordering.

Termination. Once the main loop ends, the algorithm almost finishes. Hence we shall prove the termination of this main loop. Clearly when the polynomial set F the BMS algorithm maintains turns to a Gröbner basis of $I(E)$, the current termination criterion, namely F keeps unchanged for a certain number of passes, will be satisfied. And a sufficient condition for F being a Gröbner basis is given in [19, Theorem 6].

3.5 Illustrative examples

Shape position

A toy example Katsura2 is given here to illustrate how the method based on the Wiedemann algorithm works for ideals in shape position. Consider the ideal

$$I = \langle -x_3 + 2x_2^2 + 2x_1^2 + x_3^2, -x_2 + 2x_3x_2 + 2x_2x_1, x_3 + 2x_2 + 2x_1 - 1\rangle$$

in $\mathbb{F}_{23}[x_1, x_2, x_3]$. Its Gröbner basis w.r.t. DRL is

$$\{x_1^3 + 12x_1^2 + 10x_2 + x_1, x_2^2 + 4x_1^2 + 9x_2 + 14x_1,$$
$$x_2x_1 + 15x_1^2 + 16x_2 + 18x_1, x_3 + 2x_2 + 2x_1 + 22\},$$

from which we can compute the degree of I ($D = 4$) and the basis of $\mathbb{F}_{23}[x_1, x_2, x_3]/I$ ($B = [1, x_1, x_2, x_1^2]$), and further construct the matrices T_1, T_2 and T_3.

Now we aim at computing the Gröbner basis G w.r.t. LEX. A random vector $\boldsymbol{r} = (16, 2, 18, 22)^t \in \mathbb{F}_{23}^{(4\times 1)}$ is chosen first. With the

sequence $[\langle \boldsymbol{r}, T_1^i \boldsymbol{e}\rangle : i = 0,\ldots,2D-1]$, one can obtain the first polynomial in G with the Berlekamp–Massey algorithm: $x_1^4 + 5x_1^3 + 20x_1^2 + 20x_1$. It verifies that this ideal is in shape position, hence the method is applicable. Next one can directly write the matrix H down as

$$\begin{bmatrix} 16 & 2 & 22 & 14 \\ 2 & 22 & 14 & 2 \\ 22 & 14 & 2 & 6 \\ 14 & 2 & 6 & 18 \end{bmatrix},$$

for actually all its entries have been computed in the earlier sequence. Take the polynomial $x_2 - f_2(x_1) \in G$ as in form (5) for example, the vector $\boldsymbol{b}$ can be computed as $(18,13,14,0)^t$. Solving the linear equation set $H\boldsymbol{c} = \boldsymbol{b}$, one can obtain the coefficient vector of f_2 as $(0,16,8,16)^t$, thus the corresponding polynomial in G is $x_2 + 7x_1 + 15x_1^2 + 7x_1^3$. The other polynomial $x_3 - f_3(x_1)$ can be obtained similarly. To summarize,

$$G = \{x_1^4 + 5x_1^3 + 20x_1^2 + 20x_1, x_2 + 7x_1^3 + 15x_1^2 + 7x_1, \\ x_3 + 9x_1^3 + 16x_1^2 + 11x_1 + 22\}.$$

General case

Consider the following Gröbner basis in $\mathbb{F}_{65521}[x_1,x_2]$ w.r.t. DRL $(x_1 < x_2)$

$$\begin{aligned} G_1 = \{&x_2^4 + 2x_1^3x_2 + 21x_2^3 + 11x_1x_2^2 + 4x_1^2x_2 + 22x_1^3 + 9x_2^2 \\ &+ 17x_1x_2 + 19x_1^2 + 2x_2 + 19x_1 + 5,\ x_1^2x_2^2 + 10x_2^3 \\ &+ 12x_1^2x_2 + 20x_1^3 + 21,\ x_1^4 + 15x_1^2 + 19x_1 + 3\}. \end{aligned}$$

Here $\mathbb{F}_{65521}[x_1,x_2]/\langle G_1\rangle$ is of dimension 12. Its basis, and further the multiplication matrices T_1 and T_2, can be computed accordingly.

Now we want to get the Gröbner basis G_2 of $\langle G_1\rangle$ w.r.t. LEX. With a vector

$$\boldsymbol{r} = (6757, 43420, 39830, 45356, 52762, 17712, \\ 27676, 17194, 138, 48036, 12649, 11037)^t \in \mathbb{F}_{65521}^{(12\times 1)}$$

chosen randomly, the 2-dimensional array E can be constructed. Then BMSUpdate() is applied term by term according to the LEX ordering, with the resulting Δ and F after each term shown in table 1. For example, at the term $(4,0)$, the polynomial $x_1^2 + 62681x_1 + 41493 \in F$ is not valid up to $(4,0)$. Then the delta set is updated as $\{(0,0),(1,0),(2,0)\}$, and F is reconstructed such that the new polynomial $x_1^3 + 62681x_1^2 + 35812x_1 + 18557$ is valid up to $(4,0)$.

The first polynomial in G_2

$$g_1 = x_1^4 + 15x_1^2 + 19x_1 + 3$$

is obtained at the term $(7,0)$. Clearly the method for ideals in shape position is not applicable to this example. Next BMSUpdate() is executed to compute other members of $I(E)$ according to the remaining term sequence $[x_2, x_1x_2, \ldots, x_2^2, x_1^2x_2^2, \ldots,]$, until the other polynomial in $G_2 : g_2 = x_2^3 + 7x_1^2x_2^2 + 15x_1^2x_2 + 2x_1^3 + 9$ is obtained at $(3,5)$. Now the main loop of algorithm 2 ends. Then one can easily verify that $\{g_1,g_2\} \subset G_2$ and $\dim(\mathbb{F}_{23}[x_2,x_1]/\langle g_1,g_2\rangle) = 12$, thus $G_2 = \{g_1,g_2\}$.

Here is an example where this method fails. Let $G = \{x_1^3, x_1^2x_2, x_1x_2^2, x_2^3\} \subset \mathbb{F}_{65521}[x_1,x_2]$. Then the ideal $\langle G\rangle$ is 0-dimensional with degree $D = 6$. It is easy to see that G is Gröbner basis w.r.t. both DRL and LEX. Starting from G as a Gröbner basis w.r.t. DRL, the method based on the BMS algorithm to compute the Gröbner basis w.r.t. LEX will not be able to return the correct Gröbner basis, even the base field itself is quite large and different random vectors $\boldsymbol{r}$ are tried.

4. COMPLEXITY ANALYSIS

As the main usage of algorithms for the change of ordering is to change Gröbner basis w.r.t. DRL to that w.r.t. LEX, here we also restrict the complexity analysis to cases where the target term ordering is LEX. In this paper, we assume that the multiplication matrices T_i are already computed.

4.1 Shape position

We first deal with ideals in shape position. Suppose the number of nonzero entries in T_1 is N_1. In total the Wiedemann algorithm (lines 5–7) will take $O(D(N_1 + \log(D)))$ operations to obtain the first polynomial f_1 [20]. As all the entries in the matrix H are actually already computed during the Wiedemann algorithm, its construction is free of field operations. Then, for each $i = 2,\ldots,n$, as H is a Hankel matrix, solving the linear equation set $H\boldsymbol{c} = \boldsymbol{b}_i$ only needs $O(D\log(D))$ operations [4]. As explained in section 3.1, computing $\boldsymbol{b}_i$ is equivalent to computing $\langle (T_1^t)^j \boldsymbol{r}, \boldsymbol{v}_i\rangle$, where $(T_1^t)^j$ has already been computed and $\boldsymbol{v}_i = T_i\boldsymbol{e} = \text{NormalForm}(x_i)$. Without loss of generality, we can assume that $\text{NormalForm}(x_i) = x_i$ (this is not true only if there is a linear equation $x_i + \cdots$ in the Gröbner basis G_1, and in that case we can eliminate the variable x_i). Consequently $\boldsymbol{v}_i$ is a vector with all its components equal to 0 except for one component equal to 1. Hence computing $\langle (T_1^t)^j \boldsymbol{r}, \boldsymbol{v}_i\rangle$ is equivalent to extracting some component from the vector $(T_1^t)^j\boldsymbol{r}$ and there is not additional cost. To summarize:

THEOREM 4.1. *Assume that T_1 is constructed (note that $T_2,\ldots,T_n$ are not needed). When $\deg(f_1) = D$ in algorithm 2 (line 8), the complexity of algorithm 2 is bounded by*

$$O(D(N_1 + \log(D)) + (n-1)D\log(D)) = O(D(N_1 + n\log(D))).$$

4.2 General case

Next we analyze the complexity of the BMS-based method for the general case. For the detailed description of the BMS algorithm, which is not given in this paper, readers may refer to [17]. As already explained, the computation of one value e of E can be achieved within $O(N)$ operations, where N is the maximal number of nonzero entries in matrices $T_1,\ldots,T_n$. The three steps with their complexities in the subalgorithm BMSUpdate() are:

1. checking whether every polynomial in F is valid up to $\text{Next}(\boldsymbol{u})$, which needs $O(\hat{N}D)$ operations, where $\hat{N}$ is the number of polynomials in G_2 (Note that the number of terms in every polynomial is bounded by $D+1$ because of the reduction step);

2. computing the new delta set up to $\text{Next}(\boldsymbol{u})$, which only involves integer computations and thus no field operation is needed;

3. constructing the new polynomial set F^+ such that every polynomial is valid up to $\text{Next}(\boldsymbol{u})$, which requires $O(\hat{N}D)$ operations at most.

In step 1 above, new values of E other than e may be needed for the verification. The complexity for computing them is still $O(N)$ and this is another difference from the original BMS algorithm for graded term orderings. After the update is complete, a polynomial reduction is applied to F control the size of every polynomial. This requires $O(\hat{N}\bar{N}D)$ operations, where $\bar{N}$ denotes the maximum term number of polynomials in G_2. To summarize, the total operations needed in each pass of the main loop in algorithm 2 is

$$O(N + \hat{N}D + \hat{N}\bar{N}D) = O(N + \hat{N}\bar{N}D).$$

Hence to estimate the whole complexity of the method, we only need an upper bound for the number of passes it takes in the main loop.

Table 1: Sakata

Term	Δ	F
$(0,0)$	$(0,0)$	x_1, x_2
$(1,0)$	—	$x_1 + 65437, x_2$
$(2,0)$	$(0,0),(1,0)$	$x_1^2 + 65437x_1 + 21672, x_2$
$(3,0)$	—	$x_1^2 + 62681x_1 + 41493, x_2$
$(4,0)$	$(0,0),(1,0),(2,0)$	$x_1^3 + 62681x_1^2 + 35812x_1 + 18557, x_2$
$(5,0)$	—	$x_1^3 + 30688x_1^2 + 45566x_1 + 54643, x_2$
$(6,0)$	$(0,0),(1,0),(2,0),(3,0)$	$x_1^4 + 30688x_1^3 + 20026x_1^2 + 45766x_1 + 5434, x_2$
$(7,0)$	—	g_1, x_2
$(0,1)$	—	$g_1, x_2 + 65034x_1^3 + 24330x_1^2 + 14876x_1 + 52361$
$\vdots$	$\vdots$	$\vdots$

THEOREM 4.2. *Suppose that the input ideal $I \subset \mathbb{K}[x_1,\ldots,x_n]$ is of degree D. Then the passes of the loop (lines 18–24) in algorithm 2 is bounded by $2nD$.*

PROOF. See Section 6. □

Thus the method based on the BMS algorithm for the general case requires at most $O(nD(N+\hat{N}\bar{N}D))$ field operations to finish.

5. EXPERIMENTS

The method for the shape position case has been implemented in C, while a preliminary implementation of the BMS-based method has been done in Magma. Several benchmarks are used to test the correctness and efficiency of these two methods. All the experiments were made under Scientific Linux OS release 5.5 on 8 Intel(R) Xeon(R) CPUs E5420 at 2.50 GHz with 20.55G RAM.

Table 2 illustrates performances of the implementation for the shape position case with benchmarks like Cyclic or Katsura instances, MinRank problems [13], randomly generated quadratic polynomial systems and examples coming from algebraic cryptanalysis of some curve-based cryptosystem. Instances with ideals not in shape position are marked with †, and the timings for such instances only indicate that of computing the minimal polynomial. In this table, D denotes the degree of the input ideal, and the column "Sparsity" means the percentage of nonzero entries in T_1. Timings for the computation of Gröbner bases w.r.t. DRL and the change of ordering to LEX are recorded (in seconds) for our implementation and corresponding implementations in Magma (version 2-17-1) and Singular (version 3-1-2), together with the speedup factors.

As shown by all the instances here, the multiplication matrix T_1 has a sparsity structure, even for random dense polynomial systems. Furthermore, in fact only last columns of this matrix are dense, with most of the other columns have only one nonzero component equal to 1. For matrices with such structures, we store them in a half-sparse way, that is, the sparse parts of these matrices are stored as a permutation and the others normally.

The current implementation of the algorithm for change of ordering outperforms the FGLM implementations in Magma/Singular/FGb. For example, changing the ordering to LEX for the Katsura12 instance, an ideal of degree 2^{12}, can be achieved in 26.3 seconds (1408.1 sec in Magma and 2623.5 sec in Singular respectively). It is important to note that with the new algorithm the time devoted to the change of ordering is of the same order of magnitude as the DRL Gröbner basis computation.

Table 3 illustrates the performances of the BMS-based method for the general case. As currently this method is only implemented preliminarily in Magma, only the number of field multiplications and other important parameters are recorded, instead of the timings.

Benchmarks derived from Cyclic 5 and 6 instances are used. Instances with ideals in shape position (marked with ‡) are also tested to demonstrate the generality of this method. Besides n and D denoting the number of variables and degree of the input ideal, the columns "Mat Density" and "Poly Density" denote the maximal percentage of nonzero entries in the matrices $T_1,\ldots,T_n$ and the density of resulting Gröbner bases respectively. The following 4 columns record the numbers of passes in the main loop of algorithm 2, matrix multiplications, reductions and field multiplications.

As shown in this table, the numbers of passes accord with theorem 4.2, and the number of operations is less than the original FGLM algorithm for Cyclic-like benchmarks. However, for instances with ideals in shape position, this method works but the complexity is not satisfactory. This is mainly because the resulting Gröbner bases in these cases are no longer sparse, and thus the reduction step becomes complex. The complexity may be reduced if the reduction step is handled more carefully.

6. PROOF OF THEOREM 4.2

The delta set in the BMS algorithm is determined in the following way. Given an array E, suppose F is a polynomial set valid for E up to $\boldsymbol{u}$ and the current delta set up to $\boldsymbol{u}$ is $\Delta_{\boldsymbol{u}}$. If there exists $f \in F$ such that f is not valid up to $\mathrm{Next}(\boldsymbol{u})$ and $\mathrm{Next}(\boldsymbol{u}) - \mathrm{lt}(f) \notin \Delta_{\boldsymbol{u}}$, then $\mathrm{Next}(\boldsymbol{u}) - \mathrm{lt}(f)$ is confirmed as a new term in $\Delta_{\mathrm{Next}(\boldsymbol{u})}$ [17].

This observation sheds light on which terms are indeed needed to handle in the BMS-based method on the assumption that the delta set Δ of E is known. On one hand, we have to handle the terms one of whose corresponding terms is in Δ (Criterion 1). For example, suppose $\boldsymbol{v} \in \Delta$, then the first term $\boldsymbol{u}$ such that $\boldsymbol{u} - \mathrm{lt}(f) = \boldsymbol{v}$ for some $f \in F$ has to be handled. On the other hand, we can skip those terms whose corresponding terms are not in Δ (Criterion 2), for F will be valid up to those terms automatically, otherwise the final delta set will be a wrong one. Based on these two criteria, the terms needed at most in this method are determined by the following inductive procedure.

Let G be the Gröbner basis of $I(E)$ w.r.t LEX and Δ be the delta set of E. Denote $\Delta_i = \Delta \cap \mathbb{K}[x_1,\ldots,x_i]$, $i = 1,\ldots,n$. Suppose $f \in G \cap \mathbb{K}[x_1]$ and $\deg(f) = d_1$. Then the terms needed to compute f are $P_0 = \{(j,0,\ldots,0) \in \mathbb{Z}^n_{\geq 0} : j = 0,\ldots,2d_1 - 1\}$. Set $P = \hat{P}_0$ and $Q = \Delta_1$, where $\hat{P}_0$ is the set obtained by deleting the biggest term w.r.t. LEX from P_0, These two sets P and Q will be updated from time to time in the whole procedure.

Now suppose $G \cap \mathbb{K}[x_1,\ldots,x_k]$ has been computed with updated $P, Q \subseteq \mathbb{K}[x_1,\ldots,x_k]$. Next we show the terms needed at most to obtain $G \cap \mathbb{K}[x_1,\ldots,x_{k+1}]$. For convenience, we will omit the last $n-t$ zero components for a term $\boldsymbol{u} = (u_1,\ldots,u_t,0,\ldots,0) \in \mathbb{K}[x_1,\ldots,x_t]$ if no ambiguity occurs.

Suppose $\Delta_{k+1} = \bigcup_{j=0,\ldots,m} \Delta_{k+1,j}$ for some integer m, where

$$\Delta_{k+1,j} = \Delta_{k+1} \cap \{\boldsymbol{u} : \boldsymbol{u} = (u_1,\ldots,u_k,j)\}.$$

Then we have the following results.

Table 2: Timings of the method for the shape position case from DRL to LEX

			FGb		Magma		Singular		Speedup	
Name	D	Sparsity	F_5(C)	New Algorithm	F_4	FGLM	DRL	FGLM	Magma	Singular
Katsura11	2^{11}	21.53%	4.9	3.4	18.2	178.6	632.0	328.4	52.7	96.9
Katsura12	2^{12}	21.26%	31.9	26.3	147.9	1408.1	5061.8	2623.5	53.6	99.8
Katsura13	2^{13}	19.86%	186.3	189.1	1037.2	10895.4			57.6	
Katsura14	2^{14}	19.64%	1838.9	1487.4	9599.0	87131.9			58.5	
Katsura15	2^{15}	18.52%	11456.3	12109.2						
MinR(9,6,3)	980	26.82%	1.1	0.5	6.3	22.7	137.5	38.1	43.6	73.2
MinR(9,7,4)	4116	22.95%	28.4	28.5	208.1	1360.4	4985.8	2490.3	47.7	87.4
MinR(9,8,5)	14112	19.04%	543.6	1032.8						
MinR(9,9,6)	41580	16.91%	9048.2	22171.3						
Random 11	2^{11}	21.53%	4.7	3.4	18.1	169.3	623.9	328.6	49.2	95.5
Random 12	2^{12}	21.26%	26.6	26.9	134.9	1335.8	4867.4	2581.1	49.6	95.8
Random 13	2^{13}	19.98%	146.8	193.5	949.6	10757.4	36727.0	19820.23	55.6	102.4
Random 14	2^{14}	19.64%	1000.7	1489.5	7832.4	84374.6			56.6	
Random 15	2^{15}	18.52%	6882.5	10914.02						
Weierstrass	4096	7.54%	4.0	9.0	5.8	418.3	72.4	1823.6	46.7	203.7
Edwards †	4096	3.41%	0.1	2.4	0.2	176.7	1.0	839.9	72.7	345.6
Cyclic 10 †	34940	1.00%		3586.9	>16 hrs and >16 Gig					

Table 3: Performances of the BMS-based method from DRL to LEX

Name	n	D	Mat Density	Poly Density	N. Passes	N. Matrix	N. Reduction	N. Multiplication
Cyclic5-2	2	55	4.89%	17.86%	165	318	107	$nD^{2.544}$
Cyclic5-3	3	65	8.73%	19.7%	294	704	227	$nD^{2.674}$
Cyclic5-4	4	70	10.71%	21.13%	429	1205	355	$nD^{2.723}$
Cyclic5	5	70	12.02%	21.13%	499	1347	421	$nD^{2.702}$
Cyclic6	6	156	11.46%	17.2%	1363	4464	1187	$nD^{2.781}$
Uteshev Bikker ‡	4	36	60.65%	100%	179	199	105	$nD^{2.992}$
D1 ‡	12	48	34.2%	51.02%	624	780	517	$nD^{2.874}$
Dessin2-6 ‡	6	42	46.94%	100%	294	336	205	$nD^{2.968}$

1. For $j = 0$, the terms needed by the BMS-based method are $(Q,1)$, where $(Q,i) := \{(\boldsymbol{q},i) : \boldsymbol{q} \in Q\}$.

2. For each $j = 1,\ldots,m$, the terms needed are

$$\begin{cases} (Q,i) \cup P', & i = 2j; \\ (Q,i), & i = 2j+1, \end{cases}$$

where P' is defined as follows.

(a) If there does not exist $g \in G \cap \mathbb{K}[x_1,\ldots,x_{k+1}]$ such that $\deg(g,x_{k+1}) = j$, then $P' = (P,i)$.

(b) Else suppose $\mathrm{lt}(g) = \boldsymbol{v} = (v_1,\ldots,v_k,j)$. Then

$$P' := \{\boldsymbol{u} + (0,\ldots,0,j) : \boldsymbol{u} \in \Delta_{k+1,j}\} \cup \{\boldsymbol{u} + \boldsymbol{v} : \boldsymbol{u} \in \Delta_{k+1,j}\}.$$

Furthermore, P and Q are updated as

$$Q := \{\boldsymbol{q} : (\boldsymbol{q},j) \in \Delta_{k+1,j}\}, \quad P := \{\boldsymbol{p} : (\boldsymbol{p},j) \in \hat{P}'\},$$

where $\hat{P}'$ is the set obtained by deleting the biggest term from P', similar to $\hat{P}_0$.

3. Finally when $G \cap \mathbb{K}[x_1,\ldots,x_{k+1}]$ is obtained, P is updated as the set of all terms needed for $G \cap \mathbb{K}[x_1,\ldots,x_{k+1}]$ and Q as Δ_{k+1}.

For example, figure 1 illustrates the procedure described above for the instance Cyclic5-2.

Actually the sets P and Q represent the two criteria 1 and 2 mentioned above: P is the set of terms we have to handle so that all terms in Δ are correctly added, while Q stands for those we need to handle at most according to Criterion 2. Furthermore, P is totally determined by the current $\Delta_{k+1,j}$ we have to add to Δ, and Q is determined by the latest $\Delta_{k+1,j'}$ already handled. Hence when a polynomial in $G \cap \mathbb{K}[x_1,\ldots,x_{k+1}]$ is found, say at the term $(\ldots,2j)$,

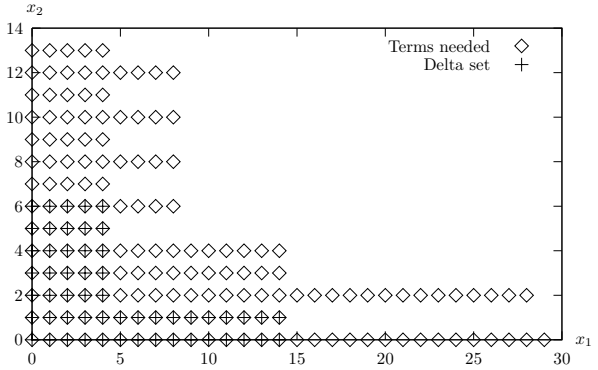

Figure 1: Delta set (+) and terms needed (◇) for Cyclic5-2

all the following $\Delta_{k+1,l}$ $(l \geq j)$ will be different from $\Delta_{k+1,j-1}$, and thus P and Q are updated accordingly.

The justification of the procedure follows naturally from the above remarks and how the terms in Δ are determined. First computation of the polynomial $f \in G \cap \mathbb{K}[x_1]$ is the same as what is done in the Berlekamp algorithm, thus at most P_0 are needed. Then P is set with one term less than P_0. This is because only $\deg(f) = d_1$, and for all the other $g \in G$, $\deg(g,x_1) < d_1$. Hence the terms needed here to get $\Delta \cap \mathbb{K}[x_1]$ have one term more than others at least. This difference can be seen from figure 1 for $x_2 = 0$ and 2. Moreover, Q is set as Δ_1.

Next for each $i = 1,\ldots,2m+1$, (Q,i) are the terms needed at most according to Criterion 2 and they are included every time. For odd $i = 2j+1$ $(j = 0,\ldots,m)$, no term is going to be added

to Δ, therefore (Q,i) are all the terms needed. While for even $i = 2j$ $(j = 1,\ldots,m)$, new terms are added to Δ. In case (a), we have

$$\Delta_{k+1,j} = \Delta_{k+1,j-1} + (0,\ldots,0,1), \tag{8}$$

thus P' here is just a translation of the previous P for $\Delta_{k+1,j-1}$. In case (b) however, the equality (8) does not hold, and P' is defined according to $\Delta_{k+1,j}$ based on both Criteria 1 and 2. Next P and Q are updated so that subsequent computation can follow correctly. Note for similar reasons to $P = \hat{P}_0$, P here is set with one term less than P'.

With the preparation above, now we are able to give the proof of Theorem 4.2.

PROOF OF THEOREM 4.2. Denote the number of terms needed to compute $G \cap \mathbb{K}[x_1,\ldots,x_i]$ by χ_i and $\Delta_i = \Delta \cap \mathbb{K}[x_1,\ldots,x_i]$ still. Clearly $\Delta = \Delta_n$. As $I \subset I(E)$, we know that Δ, the delta set of E, is a subset of the canonical basis of $\mathbb{K}[x_1,\ldots,x_n]/I$, and hence $|\Delta| \le D$. To prove the theorem, we only need to prove $2n|\Delta|$ is an upper bound.

We induce on the number of variable i of $\mathbb{K}[x_1,\ldots,x_i]$. For $i = 1$, one can easily see $\chi_1 \le 2|\Delta_1|$. Now suppose $\chi_k \le 2k|\Delta_k|$ for $k(<n)$. Next we prove $\chi_{k+1} \le 2(k+1)|\Delta_{k+1}|$.

First we ignore all the terms (Q,i) as in case (b) from all the terms needed to compute $G \cap \mathbb{K}[x_1,\ldots,x_{k+1}]$, with all the remaining terms denoted by T_{k+1}. We claim that $|T_{k+1}|$ is bounded by $(2k+1)|\Delta_{k+1}|$.

Suppose

$$\Delta_{k+1} = \bigcup_{j=0,\ldots,m} \Delta_{k+1,j}, \quad T_{k+1} = \bigcup_{l=0,\ldots,2m+1} T_{k+1,l}$$

for some integer m, where

$$\Delta_{k+1,j} = \Delta_{k+1} \cap \{\boldsymbol{u} : \boldsymbol{u} = (u_1,\ldots,u_k,j)\},$$
$$T_{k+1,l} = T_{k+1} \cap \{\boldsymbol{u} : \boldsymbol{u} = (u_1,\ldots,u_k,l)\}.$$

Then for each $\Delta_{k+1,j}$, one can see from the procedure above that $|T_{k+1,2j}|$ is bounded by either $2k|\Delta_k|$ (if $\Delta_{k+1,j}$ is before the first element in G is found, and in that case $|\Delta_{k+1,j}| = |\Delta_k|$), or $2|\Delta_{k+1,j}|$ $(\le 2k|\Delta_{k+1,j}|)$. Furthermore, $|T_{k+1,2j+1}|$ is bounded by $|\Delta_{k+1,j}|$. Hence we have

$$|T_{k+1,2j}| + |T_{k+1,2j+1}| \le (2k+1)|\Delta_{k+1,j}|,$$

which leads to $|T_{k+1}| \le (2k+1)|\Delta_{k+1}|$.

Now we only need to prove the number of all the terms (Q,i) in case (b) is bounded by $|\Delta_{k+1}|$. Suppose these cases occur at $(\boldsymbol{p}_l,l), l = i_1,\ldots,i_{m'}$. Again from the procedure, one can see that the number of terms in (Q,i) for $(\boldsymbol{p}_1,i_1)$ is bounded by $|\Delta_k|$. And after it occurs at some term $(\boldsymbol{p}_{i_l},i_l)$, the newly updated set Q_{i_l} will bound the terms occurring at $(\boldsymbol{p}_{i_{l+1}},i_{l+1})$. Then the conclusion can be proved if one notices $\Delta_k \cup (Q_{i_1},i_1) \cup \cdots \cup (Q_{i_{m'-1}},i_{m'-1}) \subseteq \Delta_{k+1}$. □

7. CONCLUDING REMARKS

Both methods proposed in this paper follow the thought of Wiedemann algorithm. That is, we take advantage of the matrix sparsity by first constructing linearly recurring relations and then finding the generators for these relations with the (generalized) Berlekamp–Massey algorithm. Multiplication matrices in the FGLM algorithm serve as a bridge between the change of ordering and linearly recurring relations.

The BMS algorithm itself, as a multi-dimensional generalization of the Berlekamp–Massey algorithm, is worth studying. We hope that this paper is just a first step for the study of this algorithm. Several problems concerning it are still unsolved and left as future works: the complete characterization of its termination criteria, the probability for $I(E) = I$, what to do when $I(E) \neq I$, and further improvement of the algorithm.

Moreover, the sparsity of multiplication matrices is now demonstrated by several benchmarks. Could we express the sparsity of the matrices as $O(D^\alpha)$ with $\alpha < 2$, it would give immediately a better complexity for the change of ordering.

8. REFERENCES

[1] A. Basiri and J.-C. Faugère. Changing the ordering of Gröbner bases with LLL: case of two variables. In *Proceedings of ISSAC 2003*, pages 23–29. ACM, 2003.

[2] E. Becker, T. Mora, M. Marinari, and C. Traverso. The shape of the Shape Lemma. In *Proceedings of ISSAC 1994*, pages 129–133. ACM, 1994.

[3] T. Becker, V. Weispfenning, and H. Kredel. *Gröbner Bases: a Computational Approach to Commutative Algebra*. Graduate Texts in Mathematics. Springer, New York, 1993.

[4] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun. Fast solution of toeplitz systems of equations and computation of Padé approximants. *Journal of Algorithms*, 1(3):259–295, 1980.

[5] B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In *Multidimensional Systems Theory*, pages 184–232. Reidel, Dordrecht, 1985.

[6] J. Buchmann, A. Pyshkin, and R.-P. Weinmann. A zero-dimensional Gröbner basis for AES-128. In M. Robshaw, editor, *Fast Software Encryption*, volume 4047 of *LNCS*, pages 78–88. Springer, Berlin / Heidelberg, 2006.

[7] S. Collart, M. Kalkbrener, and D. Mall. Converting bases with the Gröbner walk. *Journal of Symbolic Computation*, 24(3–4):465–469, 1997.

[8] D. A. Cox, J. B. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra (2nd edn.)*. Undergraduate Texts in Mathematics. Springer, New York, 1997.

[9] X. Dahan, X. Jin, M. Moreno Maza, and E. Schost. Change of order for regular chains in positive dimension. *Theoretical Computer Science*, 392:37–65, 2008.

[10] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, 1999.

[11] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *Proceedings of ISSAC 2002*, pages 75–83. ACM, 2002.

[12] J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.

[13] J.-C. Faugère, M. Safey El Din, and P.-J. Spaenlehauer. Computing loci of rank defects of linear matrices using Gröbner bases and applications to cryptology. In *Proceedings of ISSAC 2010*, pages 257–264. ACM, 2010.

[14] E. Jonckheere and C. Ma. A simple Hankel interpretation of the Berlekamp–Massey algorithm. *Linear Algebra and its Applications*, 125:65–76, 1989.

[15] P. Loustaunau and E. York. On the decoding of cyclic codes using Gröbner bases. *Applicable Algebra in Engineering, Communication and Computing*, 8(6):469–483, 1997.

[16] C. Pascal and E. Schost. Change of order for bivariate triangular sets. In *Proceedings of ISSAC 2006*, pages 277–284. ACM, 2006.

[17] K. Saints and C. Heegard. Algebraic-geometric codes and multidimensional cyclic codes: a unified theory and algorithms for decoding using Gröbner bases. *IEEE Transactions on Information Theory*, 41(6):1733–1751, 2002.

[18] S. Sakata. Finding a minimal set of linear recurring relations capable of generating a given finite two-dimensional array. *Journal of Symbolic Computation*, 5(3):321–337, 1988.

[19] S. Sakata. Extension of the Berlekamp–Massey algorithm to N dimensions. *Information and Computation*, 84(2):207–239, 1990.

[20] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1):54–62, 1986.