Journal of Symbolic Computation 80 (2017) 538–569

Contents lists available at ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Sparse FGLM algorithms

Jean-Charles Faugère [a], Chenqi Mou [b,a]

[a] Sorbonne Universités, UPMC Univ Paris 06, CNRS, INRIA, Laboratoire d'Informatique de Paris 6 (LIP6), Équipe PolSys, 4 place Jussieu, 75005 Paris, France
[b] LMIB – School of Mathematics and Systems Science, Beihang University, Beijing 100191, China

ARTICLE INFO

Article history:
Received 18 February 2016
Accepted 5 July 2016
Available online 21 July 2016

Keywords:
Gröbner bases
Change of ordering
Zero-dimensional ideal
Sparse matrix
Wiedemann algorithm
BMS algorithm

ABSTRACT

Given a zero-dimensional ideal $I \subset \mathbb{K}[x_1, \ldots, x_n]$ of degree D, the transformation of the ordering of its Gröbner basis from DRL to LEX is a key step in polynomial system solving and turns out to be the bottleneck of the whole solving process. Thus it is of crucial importance to design efficient algorithms to perform the change of ordering.

The main contributions of this paper are several efficient methods for the change of ordering which take advantage of the sparsity of multiplication matrices in the classical FGLM algorithm. Combining all these methods, we propose a deterministic top-level algorithm that automatically detects which method to use depending on the input. As a by-product, we have a fast implementation that is able to handle ideals of degree over 60000. Such an implementation outperforms the Magma and Singular ones, as shown by our experiments.

First for the shape position case, two methods are designed based on the Wiedemann algorithm: the first is probabilistic and its complexity to complete the change of ordering is $O(D(N_1+n\log(D)^2))$, where N_1 is the number of nonzero entries of a multiplication matrix; the other is deterministic and computes the LEX Gröbner basis of $\sqrt{I}$ via Chinese Remainder Theorem. Then for the general case, the designed method is characterized by the Berlekamp–Massey–Sakata algorithm from Coding Theory to handle the multi-dimensional linearly recurring relations. Complexity analyses of all proposed methods are also provided.

Furthermore, for generic polynomial systems, we present an explicit formula for the estimation of the sparsity of one main multiplication matrix, and prove that its construction is free. With the

E-mail addresses: Jean-Charles.Faugere@inria.fr (J.-C. Faugère), Chenqi.Mou@buaa.edu.cn (C. Mou).

http://dx.doi.org/10.1016/j.jsc.2016.07.025

asymptotic analysis of such sparsity, we are able to show that for generic systems the complexity above becomes $O(\sqrt{6/n\pi}D^{2+\frac{n-1}{n}})$.

1. Introduction

1.1. Motivation

Gröbner basis is an important tool in computational ideal theory (Buchberger, 1985; Cox et al., 1998; Becker et al., 1993), especially for polynomial system solving. For a given ideal and term ordering, the Gröbner basis of this ideal with respect to (w.r.t.) the term ordering is a set of generators with good properties, such that manipulation of the ideal can be achieved with these generators.

The term ordering plays an important role in the theory of Gröbner bases. It is well-known that Gröbner bases w.r.t. different term orderings are also different and possess different theoretical and computational properties. For example, Gröbner bases w.r.t. the lexicographical ordering (LEX) have good algebraic structures and are convenient to use for polynomial system solving, while those w.r.t. the degree reverse lexicographical ordering (DRL) are computationally easy to obtain. Therefore, the common strategy to solve a polynomial system is to first compute the Gröbner basis of the ideal defined by the system w.r.t. DRL, change its ordering to LEX, and perhaps further convert the LEX Gröbner basis to triangular sets (Lazard, 1992) or Rational Univariate Representation (Rouillier, 1999). That is one of the main usages of algorithms for the change of ordering.

However, with the computation of Gröbner bases greatly enhanced recently (Faugère, 1999, 2002), the step to change the orderings of Gröbner bases has become the bottleneck of the whole solving process (see Section 6). Hence it is of crucial significance to design efficient algorithms for the change of ordering. The purpose of this paper is precisely to provide such efficient algorithms.

Furthermore, some practical problems can be directly modeled as the change of ordering of Gröbner bases. For example, the Gröbner basis of an ideal derived from the AES-128 cryptosystem w.r.t. a certain term ordering (other than LEX) has been obtained (Buchmann et al., 2006), and it may lead to a successful cryptanalysis on this system if one is able to convert its term ordering to LEX. And the decoding of some cyclic codes can also be regarded as a problem of changing the term orderings (Loustaunau and York, 1997).

1.2. Related works

Several algorithms for the change of ordering have already existed, for example the FGLM algorithm for the zero-dimensional case (Faugère et al., 1993) and the Gröbner walk for the general case (Collart et al., 1997). Similar algorithms have also been proposed to change the orderings of triangular sets (Pascal and Schost, 2006; Dahan et al., 2008) or using the LLL algorithm (Basiri and Faugère, 2003) in the bivariate case.

Among them, the FGLM algorithm, only applicable to the zero-dimensional case, is an efficient one. The number of field operations it needs to complete the change of ordering is $O(nD^3)$, where n is the number of variables, and D is the degree of the given ideal $I \subset \mathbb{K}[x_1, \ldots, x_n]$. Its efficiency may be due to the fact that it reduces the problem of change of ordering to linear algebra operations. Such a connection is achieved through the multiplication matrix T_i $(i = 1, \ldots, n)$ used in this algorithm, which represents the multiplication by x_i in the quotient ring $\mathbb{K}[x_1, \ldots, x_n]/I$ viewed as a vector space. These matrices are sparse, even when the input polynomial system is dense (see Section 5). And in this paper we take advantage of this sparsity structure to obtain fast FGLM algorithms with good complexity and performances.

1.3. Our contributions and paper organization

We first study the particular but important case when the zero-dimensional ideal I is in shape position. Two methods based on the Wiedemann algorithm are proposed to compute the Gröbner bases of I or $\sqrt{I}$ w.r.t. LEX. They both make use of the sparsity by constructing the linearly recurring sequence

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

where $\boldsymbol{r}$ is a random vector and $\boldsymbol{e} = (1, 0, \dots)^t$ is the vector representing $\mathbf{1}$ in $\mathbb{K}[x_1, \dots, x_n]/I$. 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 $\deg(f_1) = D$, and it can be computed by applying the Berlekamp–Massey algorithm (Wiedemann, 1986; Kaltofen and Pan, 1991). Furthermore, we show how to recover efficiently the other polynomials in the Gröbner basis by solving structured (Hankel) linear systems. Hence, we are able to complete the first method for the change of ordering to LEX for ideals in shape position with complexity $O(D(N_1 + n\log(D)^2))$, 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. This method is a probabilistic one and is summarized as Algorithm 3 named "ShapePositionProbabilistic()" (Section 3.1).

The other method for the shape position case uses the deterministic Wiedemann algorithm, which can always return the correct univariate polynomial in the Gröbner basis w.r.t. LEX. Making use of the Chinese Remainder Theorem, this method adapts and extends the previous one to recover the Gröbner basis of $\sqrt{I}$, instead of I. Thus it is suitable to those problems where the zeros, instead of the multiplicities, are of interest. For ideals in shape position, this deterministic method can always return the Gröbner basis of $\sqrt{I}$ w.r.t. LEX with the complexity $O(D(N_1 + D\log(D)^2 + nD + R))$, where $R = 0$ if $\mathbb{K}$ has characteristic 0 and $R = \log(q/p)$ if $\mathbb{K}$ has characteristic $p > 0$ and $|\mathbb{K}| = q$. This deterministic method is summarized as Algorithm 4 named "ShapePositionDeterministic()" (Section 3.2).

We also briefly discuss how to apply an incremental variant of the Wiedemann algorithm to compute the univariate polynomial, which is of special importance among all the polynomials in the Gröbner basis. Such an variant has a complexity sensitive to the output, namely the degree of the univariate polynomial, and is efficient when this degree is small.

Then for general ideals to which the methods above may be no longer applicable, we follow the idea above by generalizing the linearly recurring sequence to an n-dimensional mapping

$$E : (s_1, \dots, s_n) \longmapsto \langle \boldsymbol{r}, T_1^{s_1} \cdots T_n^{s_n} \boldsymbol{e}\rangle.$$

The minimal set of generating polynomials (w.r.t. a term ordering) 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 (Sakata, 1988, 1990). With modifications of the BMS algorithm, we design a method to change the ordering in the general case. The complexity of this algorithm is $O(nD(N + \hat{N}\overline{N}D))$, where N is the maximal number of nonzero entries in matrices $T_1, \dots, T_n$, while $\hat{N}$ and $\overline{N}$ are respectively the number of polynomials and the maximal term number of all polynomials in the resulting Gröbner basis. This method is also probabilistic and is summarized as Algorithm 5 named "GeneralBMSbased()" (Section 4).

According to the criterion proposed in the original FGLM algorithm, one is able to check whether the polynomial set computed by either algorithm above is the target Gröbner basis with little efforts (see Section 2.1 or Faugère et al. (1993) for the details). To conclude, our overall strategy in the algorithms above is to "guess" the target Gröbner basis in an efficient way with probability, and then check the correctness by applying this criterion.

Combining all these algorithms above, we are able to present the following deterministic top-level algorithm (Algorithm 1). Thanks to the feasibility in each algorithm to test whether the computed polynomial set is the correct Gröbner basis, this top-level algorithm will automatically select which algorithm to use according to the input, until the original FGLM one is called if all these algorithms fail. It is also a deterministic algorithm, though both the Wiedemann algorithm and the BMS-based method will introduce randomness and probabilistic behaviors to the individual algorithms.

The efficiency of the proposed methods is verified by experiments. The current implementation outperforms those of the FGLM algorithm in Magma and Singular. Take a randomly generated quadratic polynomial system of 13 variables for example, it generates an ideal in shape position of

Algorithm 1: Top-level algorithm $G_2 := \mathsf{TopLevel}(G_1, <_1)$.

```
    Input: $G_1$, Gröbner basis of a 0-dimensional ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ w.r.t. $<_1$
    Output: $G_2$, Gröbner basis of $I$ or $\sqrt{I}$ w.r.t. LEX
 1  $G_2 :=$ ShapePositionProbabilistic$(G_1, <_1)$                    // Section 3.1
 2  if $G_2 \neq$ Fail then
 3      return $G_2$
 4  else
 5      $G_2 :=$ ShapePositionDeterministic$(G_1, <_1)$                // Section 3.2
 6      if $G_2 \neq$ Fail then
 7          return $G_2$
 8      else
 9          $G_2 :=$ GeneralBMSbased$(G_1, <_1)$                       // Section 4
10          if $G_2 \neq$ Fail then
11              return $G_2$
12          else
13              return FGLM$(G_1, <_1)$
14          end
15      end
16  end
```

degree 8192. For such an ideal, the change of ordering to LEX can be achieved in 80.8 seconds: this is about 132 and 244 times faster than the corresponding MAGMA and SINGULAR functions. As shown in Table 2, zero-dimensional ideals over a prime field of degree greater than 60000 are now tractable. Details of the experimental results are presented in Section 6.

Furthermore, the performances of these methods are heavily dependent on the sparsity of the multiplication matrices, especially T_1 for the shape position case. In general we assume the multiplication matrices known. However, for generic polynomial systems consisting of n-variate polynomials of degree d, the sparsity of T_1 is investigated, and we are able to give an explicit formula to compute the number of dense columns in T_1 and show that indeed the construction of T_1 is free. These results furnish a complete complexity analysis of the proposed method for generic polynomial systems. Then with an asymptotic analysis of the number of dense columns as d tends to $+\infty$, we show that the complexity of the probabilistic method for the shape position case becomes $O(\sqrt{6/n\pi}\, D^{2+\frac{n-1}{n}})$ for generic systems. Such simplified complexity is better than that of FGLM with smaller constant and exponent. This part of contribution on sparsity of T_1 is presented in Section 5.

1.4. What is new

To be self-contained, this paper also includes results obtained in Faugère and Mou (2011) in a refined way for description. However, several original extensions have also been presented here, making the discussion on this subject more comprehensive: (1) For ideals in shape position, one new algorithm is proposed based on the deterministic Wiedemann algorithm. Compared with the previous probabilistic one, this algorithm becomes deterministic and aims at the Gröbner basis of the radical of the input ideal. (2) The multiplication matrices are assumed known in Faugère and Mou (2011), but here for the multiplication matrix T_1 which is of special importance, its sparsity, together with the asymptotic behaviors, and construction cost are analyzed for generic polynomial systems. Such a study furnishes a complete understanding on the complexity of the change of ordering for generic systems. (3) Descriptions for the BMS algorithm for the LEX ordering are added in Section 4.2, and the proof of Theorem 4.1 is further simplified via introduction of known results in the literature. (4) Experimental results are updated.

The authors would like to remark that this paper is different from another paper Faugère et al. (2014) on the change of ordering of Gröbner bases in the sense that we mainly study how to design sparse algorithms for change of ordering and do not concentrate much on the construction of multiplication matrices except for the generic case, while the paper Faugère et al. (2014) mainly studies how to compute these multiplication matrices for all the cases even when they are not sparse.

2. Backgrounds: FGLM and BMS algorithms

Let $\mathbb{K}[x_1,\ldots,x_n]$ be the n-variate polynomial ring over a field $\mathbb{K}$, with variables ordered as $x_1 < \cdots < x_n$. Suppose G_1 is the Gröbner basis of a 0-dimensional ideal $I \subset \mathbb{K}[x_1,\ldots,x_n]$ w.r.t. a term ordering $<_1$. Given another term ordering $<_2$, one wants to compute the Gröbner basis G_2 of I w.r.t. it. Denote by D the degree of I, that is, the dimension of $\mathbb{K}[x_1,\ldots,x_n]/I$ as a vector space. These notations are fixed hereafter in this paper.

2.1. FGLM algorithm

The FGLM algorithm is one to perform the change of ordering of Gröbner bases of 0-dimensional ideals efficiently (Faugère et al., 1993). The reason why it is fast may be due to the idea that it reduces the change of ordering to linear algebra operations in the quotient ring $\mathbb{K}[x_1,\ldots,x_n]/I$. Such a reduction is realized in the following way.

First one computes the canonical basis of $\mathbb{K}[x_1,\ldots,x_n]/\langle G_1\rangle$ and orders its elements according to $<_1$. Let $B = [\epsilon_1,\ldots,\epsilon_D]$ be the ordered basis. Then ϵ_1 will always equal $\mathbf{1}$, for $<_1$ is a term ordering. Given a variable x_i, for each element ϵ_j in B, one can compute the normal form of $\epsilon_j x_i$ w.r.t. G_1, denoted by $\mathrm{NormalForm}(\epsilon_j x_i)$. This normal form, viewed as an element in $\mathbb{K}[x_1,\ldots,x_n]/\langle G_1\rangle$, can be further written as a linear combination of B. Writing the coefficients as a column vector, one can construct a $D \times D$ matrix T_i by adjoining all the column vectors for $j = 1,\ldots,D$. This matrix is called the *multiplication matrix* of x_i. It is not hard to verify that all T_i commute: $T_i T_j = T_j T_i$ for $i, j = 1,\ldots,n$.

Next one handles all the terms in $\mathbb{K}[x_1,\ldots,x_n]$ one by one following $<_2$. For each term $\boldsymbol{x}^{\boldsymbol{s}}$ with $\boldsymbol{s} = (s_1,\ldots,s_n)$, its coordinate vector w.r.t. B can be computed by $\boldsymbol{v}_{\boldsymbol{s}} = T_1^{s_1}\cdots T_n^{s_n}\boldsymbol{e}$, where $\boldsymbol{e} = (1,0,\ldots,0)^t$ is the coordinate vector of $\mathbf{1}$. Then criteria proposed in the FGLM algorithm guarantee that once a linear dependency of the coordinate vectors of computed terms

$$\sum_{\boldsymbol{s}\in S} c_{\boldsymbol{s}} \boldsymbol{v}_{\boldsymbol{s}} = 0 \tag{1}$$

is found, a polynomial $f \in G_2$ can be directly derived in the following form

$$f = \boldsymbol{x}^{\boldsymbol{l}} + \sum_{\boldsymbol{s}\in S, \boldsymbol{s}\neq \boldsymbol{l}} \frac{c_{\boldsymbol{s}}}{c_{\boldsymbol{l}}} \boldsymbol{x}^{\boldsymbol{s}}, \tag{2}$$

where $\boldsymbol{x}^{\boldsymbol{l}}$ is the leading term of f w.r.t. $<_2$ (denoted by $\mathrm{lt}(f)$) (Faugère et al., 1993).

As can be seen now, all one needs to do to obtain the Gröbner basis G_2 is computing the coordinate vector of each term one by one, and checking whether a linear dependency of these vectors occurs after a new vector is computed, which can be realized by maintaining an echelon form of the matrix whose columns are coordinate vectors of previously computed terms. These steps are merely matrix manipulations from linear algebra. A trivial upper bound for the number of terms to consider is $D+1$ because of the vector size.

2.2. BMS algorithm

The BMS algorithm from Coding Theory is a decoding algorithm to find the generating set of the error locator ideal in algebraic geometry codes (Sakata, 1988, 1990; Saints and Heegard, 2002). From a more mathematical point of view, it computes the set of minimal polynomials (w.r.t. a term ordering $<$) of a linearly recurring relation generated by a given multi-dimensional array. It is a generalization of the Berlekamp–Massey algorithm, which is applied to Reed–Solomon codes to find the generating error locator polynomial, or mathematically the minimal polynomial of a linearly recurring sequence.

The BMS algorithm, without much modification, can also be extended to a more general setting of order domains (Cox et al., 1998; Høholdt et al., 1998). Combining with the Feng–Rao majority voting algorithm (Feng and Rao, 1993), this algorithm can often decode codes with more than $(d_{\min} - 1)/2$

errors if the error locations are general (Bras-Amorós and O'Sullivan, 2006), where $d_{\min}$ is the minimal distance. Next a concise description of the BMS algorithm is given, focusing on its mathematical meanings.

As a vector $\boldsymbol{u} = (u_1, \dots, u_n) \in \mathbb{Z}^n_{\geq 0}$ and a term $\boldsymbol{x}^{\boldsymbol{u}} = x_1^{u_1} \cdots x_n^{u_n} \in \mathbb{K}[x_1, \dots, x_n]$ are 1–1 corresponding, usually we do not distinguish one from the other. A mapping $E : \mathbb{Z}^n_{\geq 0} \longrightarrow \mathbb{K}$ is called an *n-dimensional array*. In Coding Theory, the array E is usually a syndrome array determined by the error word (Saints and Heegard, 2002). Besides the term ordering, we define the following partial ordering: for two terms $\boldsymbol{u} = (u_1, \dots, u_n)$ and $\boldsymbol{v} = (v_1, \dots, v_n)$, we say that $\boldsymbol{u} \prec \boldsymbol{v}$ if $u_i \leq v_i$ for $i = 1, \dots, n$.

Definition 2.1. Given a polynomial $f = \sum_{\boldsymbol{s}} f_{\boldsymbol{s}} x^{\boldsymbol{s}} \in \mathbb{K}[x_1, \dots, x_n]$, an n-dimensional mapping E is said to satisfy the *n-dimensional linearly recurring relation* with the *characteristic polynomial* f if

$$\sum_{\boldsymbol{s}} f_{\boldsymbol{s}} E_{\boldsymbol{s}+\boldsymbol{r}} = 0, \quad \forall \boldsymbol{r} \succ 0. \tag{3}$$

The set of all characteristic polynomials of the n-dimensional linearly recurring relation for the array E forms an ideal, denoted by $I(E)$. Again in the setting of decoding when E is a syndrome array, this ideal is called the *error locator ideal* for E, and its elements are called *error locators*. The definition of $I(E)$ used here in this paper follows Saints and Heegard (2002), and one can easily see that this definition is equivalent to that in Cox et al. (1998) by Saints and Heegard (2002, Thm. 23).

Furthermore, the set of minimal polynomials for $I(E)$ w.r.t. $<$, which the BMS algorithm computes, is actually the Gröbner basis of $I(E)$ w.r.t. $<$ (Sakata, 1990, Lem. 5). The canonical basis of $\mathbb{K}[x_1, \dots, x_n]/I(E)$ is also called the *delta set* of E, denoted by $\Delta(E)$. The term "delta set" comes from the property that if $\boldsymbol{u} \in \mathbb{Z}^n_{\geq 0}$ is contained in $\Delta(E)$, then $\Delta(E)$ also contains all elements $\boldsymbol{v} \in \mathbb{Z}^n_{\geq 0}$ such that $\boldsymbol{v} \prec \boldsymbol{u}$.

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 the given term ordering $<$. A polynomial f with $\operatorname{lt}(f) = \boldsymbol{s}$ is said to be *valid for E up to* $\boldsymbol{u}$ if either $\boldsymbol{u} \not\succ \boldsymbol{s}$ or

$$\sum_{\boldsymbol{t}} f_{\boldsymbol{t}} E_{\boldsymbol{t}+\boldsymbol{r}} = 0, \quad \forall \boldsymbol{r}\ (0 \prec \boldsymbol{r} \preceq \boldsymbol{u} - \boldsymbol{s}). \tag{4}$$

E may be omitted if no ambiguity occurs. A polynomial set is said to be valid up to $\boldsymbol{u}$ if each of its polynomials is so.

Similarly to FGLM, the BMS algorithm also handles terms in $\mathbb{K}[x_1, \dots, x_n]$ one by one according to $<$, so that the polynomial set F it maintains is valid up to the new term. Suppose that F is valid up to some term $\boldsymbol{u}$. When the next term of $\boldsymbol{u}$ w.r.t. $<$, denoted by $\operatorname{Next}(\boldsymbol{u})$, is considered, the BMS algorithm will update F so that F keeps valid up to $\operatorname{Next}(\boldsymbol{u})$. Meanwhile, terms determined by $\operatorname{Next}(\boldsymbol{u})$ are also tested whether they are members of $\Delta(E)$. Therefore, more and more terms will be verified in $\Delta(E)$ as the BMS algorithm proceeds. The set of verified terms in $\Delta(E)$ after the term $\boldsymbol{u}$ is called the *delta set up to* $\boldsymbol{u}$ and denoted by $\Delta(\boldsymbol{u})$. Then we have

$$\Delta(\mathbf{1}) \subset \cdots \subset \Delta(\boldsymbol{u}) \subset \Delta(\operatorname{Next}(\boldsymbol{u})) \subset \cdots \subset \Delta(E).$$

After a certain number of terms are handled, F and $\Delta(\boldsymbol{u})$ will grow to the Gröbner basis of $I(E)$ and $\Delta(E)$ respectively.

Next only the outlines of the update procedure mentioned above, which is also the main part of the BMS algorithm, are presented as Algorithm 2 for convenience of later use. More details will also be provided in Section 4. One may refer to Saints and Heegard (2002), Cox et al. (1998) for a detailed description. In Algorithm 2 below, the polynomial set G, called the *witness set*, is auxiliary and will not be returned with F at the end of the BMS algorithm.

3. Shape position case: probabilistic, deterministic and incremental algorithms

In this section, the case when the ideal I is in shape position is studied.

```
Algorithm 2: $(F^+, G^+) := \mathsf{BMSUpdate}(F, G, \mathsf{Next}(\boldsymbol{u}), E)$.
Input:
    $F$, a minimal polynomial set valid up to $\boldsymbol{u}$;
    $G$, a witness set up to $\boldsymbol{u}$;
    $\mathsf{Next}(\boldsymbol{u})$, a term;
    $E$, an $n$-dimensional array up to $\mathsf{Next}(\boldsymbol{u})$.
Output:
    $F^+$, a minimal polynomial set valid up to $\mathsf{Next}(\boldsymbol{u})$;
    $G^+$, a witness set up to $\mathsf{Next}(\boldsymbol{u})$.
1. Test whether every polynomial in $F$ is valid up to $\mathsf{Next}(\boldsymbol{u})$
2. Update $G^+$ and compute the new delta set up to $\mathsf{Next}(\boldsymbol{u})$ accordingly
3. Construct new polynomials in $F^+$ such that they are valid up to $\mathsf{Next}(\boldsymbol{u})$
```

Definition 3.1. An ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ is said to be *in shape position* if its Gröbner basis w.r.t. LEX is of the following form

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

One may easily see that I here is 0-dimensional and $\deg(f_1) = D$. Such ideals take a large proportion in all the consistent ideals and have been well studied and applied (Becker et al., 1994; Rouillier, 1999). The special structure of their Gröbner bases enables us to design specific and efficient methods to change the term ordering to LEX. In the following, methods designed for different purposes, along with their complexity analyses, are exploited.

Throughout this section, we assume that the multiplication matrix T_1 is nonsingular. Otherwise, one knows by the Stickelberger's theorem (cf. Rouillier, 1999, Thm. 2.1) that $x_1 = 0$ will be a root of the univariate polynomial in the Gröbner basis of I w.r.t. LEX, and usually the polynomial system can be further simplified.

3.1. *Probabilistic algorithm to compute Gröbner basis of the ideal*

3.1.1. *Algorithm description*

Given a 0-dimensional ideal I, if the univariate polynomial $f_1(x_1)$ in its Gröbner basis w.r.t. LEX is of degree D, then we know that I is in shape position.

The way to compute such a univariate polynomial is the Wiedemann algorithm. Consider now the following linearly recurring sequence

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

where $\boldsymbol{r}$ is a randomly generated vector in $\mathbb{K}^{(D\times 1)}$, T_1 is the multiplication matrix of x_1, $\boldsymbol{e}$ is the coordinate vector of $\mathbf{1}$ w.r.t. the canonical basis of $\mathbb{K}[x_1, \dots, x_n]/I$, and $\langle \cdot, \cdot \rangle$ takes the inner product of two vectors. It is not hard to see that the minimal polynomial $\tilde{f}_1$ of the sequence s is a factor of f_1. As D is always a bound on the size of the linearly recurring sequence, the Berlekamp–Massey algorithm can be applied to the sequence s to compute $\tilde{f}_1$. Furthermore, if $\deg(\tilde{f}_1) = D$, then $\tilde{f}_1 = f_1$ and I can be verified in shape position.

Suppose that $\deg(\tilde{f}_1) = D$ holds and f_i in (5) is of the form $f_i = \sum_{k=0}^{D-1} c_{i,k} x_1^k$ for $i = 2, \dots, n$. Then computing the whole Gröbner basis of I w.r.t. LEX reduces to determining all the unknown coefficients $c_{i,k}$. Before we show how to recover them, some basic results about linearly recurring sequences are recalled.

Definition 3.2. Let $s = [s_0, s_1, s_2, \cdots]$ be a sequence of elements in $\mathbb{K}$ and d an integer. Then the $d \times d$ *Hankel matrix* is defined as

$$H_d(s) = \begin{bmatrix} s_0 & s_1 & s_2 & \cdots & s_{d-1} \\ s_1 & s_2 & s_3 & \cdots & s_d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{d-1} & s_d & s_{d+1} & \cdots & s_{2d-2} \end{bmatrix}.$$

Theorem 3.1 *(Jonckheere and Ma, 1989; Kaltofen and Pan, 1991). Let $s = [s_0, s_1, s_2, \cdots]$ be a linearly recurring sequence. Then the minimal polynomial $M^{(s)}(x) = \sum_{i=0}^{d} m_i x^i$ of the sequence s is such that:*

(i) *$d = \mathrm{rank}(H_d(s)) = \mathrm{rank}(H_i(s))$ for all $i > d$;*
(ii) *$\ker(H_{d+1}(s))$ is a vector space of dimension 1 generated by $(m_0, m_1, \dots, m_d)^t$.*

For each $i = 2, \dots, n$, as $x_i - \sum_{k=0}^{D-1} c_{i,k} x_1^k \in I$, one has $\mathrm{NormalForm}(x_i - \sum_{k=0}^{D-1} c_{i,k} x_1^k) = 0$, and thus

$$\boldsymbol{v}_i := T_i \boldsymbol{e} = \sum_{k=0}^{D-1} c_{i,k} \cdot T_1^k \boldsymbol{e}.$$

Multiplying T_1^j and taking the inner product with a random vector $\boldsymbol{r}$ to both hands for $j = 1, \dots, D-1$, one can further construct D linear equations

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

With $c_{i,k}$ considered as unknowns, the coefficient matrix H with entries $\langle \boldsymbol{r}, T_1^{k+j}\boldsymbol{e} \rangle$ is indeed a $D \times D$ Hankel matrix, and thus invertible by Theorem 3.1. Furthermore, the linear equation set (7) with the Hankel matrix H can be efficiently solved (Brent et al., 1980; Kaltofen and Pan, 1991). All the solutions of these linear systems for $i = 2, \dots, n$ will lead to the Gröbner basis we want to compute.

The method above is summarized in the following algorithm (Algorithm 3), whose termination and correctness are direct results based on previous discussions. The subfunction BerlekampMassey() is the Berlekamp–Massey algorithm, which takes a sequence over $\mathbb{K}$ as input and returns the minimal polynomial of this sequence (Wiedemann, 1986).

Algorithm 3: Shape position (probabilistic) $G_2 :=$ ShapePositionProbabilistic($G_1, <_1$).

```
Input: $G_1$, Gröbner basis of a 0-dimensional ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ w.r.t. $<_1$
Output: $G_2$, Gröbner basis of $I$ w.r.t. LEX if the polynomial returned by BerlekampMassey() is of degree $D$; Fail,
        otherwise.
1  Compute the canonical basis of $\mathbb{K}[x_1, \dots, x_n]/\langle G_1 \rangle$ and multiplication matrices $T_1, \dots, T_n$;
2  $\boldsymbol{e} := (1, 0, \dots, 0)^t \in \mathbb{K}^{(D\times 1)}$;
3  Choose $\boldsymbol{r}_0 = \boldsymbol{r} \in \mathbb{K}^{(D\times 1)}$ randomly;
4  for $i = 1, \dots, 2D-1$ do
5      $\boldsymbol{r}_i := (T_1^t)\boldsymbol{r}_{i-1}$;
6  end
7  Generate the sequence $s := [\langle \boldsymbol{r}_i, \boldsymbol{e} \rangle : i = 0, \dots, 2D-1]$;
8  $f_1 :=$ BerlekampMassey($s$);
9  if $\deg(f_1) = D$ then
10     $H := H_D(s)$                                   // Construct the Hankel matrix
11     for $i = 2, \dots, n$ do
12         $\boldsymbol{b} := (\langle \boldsymbol{r}_j, T_i \boldsymbol{e} \rangle : j = 0, \dots, D-1)^t$;
13         Compute $\boldsymbol{c} = (c_1, \dots, c_D)^t := H^{-1}\boldsymbol{b}$;
14         $f_i := \sum_{k=0}^{D-1} \boldsymbol{c}_{k+1} x_1^k$;
15     end
16     return $[f_1, x_2 - f_2, \dots, x_n - f_n]$;
17 else
18     return Fail;
19 end
```

Remark 3.1. As can be seen from the description of Algorithm 3, such a method is a probabilistic one. That is to say, it can return the correct Gröbner basis w.r.t. LEX with probabilities, and may also fail even when I is indeed in shape position.

3.1.2. Complexity

In this complexity analysis and others to follow, we assume that the multiplication matrices are all known and neglect their construction cost.

Suppose that the number of nonzero entries in T_1 is N_1. The Wiedemann algorithm (both construction of the linearly recurring sequence and computation of its minimal polynomial with the Berlekamp–Massey algorithm) will take $O(D(N_1 + \log(D)))$ field operations to return the minimal polynomial $\tilde{f}_1$ (Wiedemann, 1986).

Next we show how the linear system (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 in (6) and (7)

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

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

First, as each entry $\langle \boldsymbol{r}, T_1^{k+j}\boldsymbol{e}\rangle$ of the Hankel matrix H can be extracted from the sequence (6), the construction of H is free of operations. What is left now is the computation of $\langle (T_1^t)^j\boldsymbol{r}, \boldsymbol{v}_i\rangle$, where $(T_1^t)^j\boldsymbol{r}$ has already been computed and $\boldsymbol{v}_i = T_i\boldsymbol{e} = \mathrm{NormalForm}(x_i)$. Without loss of generality, we can assume that $\mathrm{NormalForm}(x_i) = x_i$ (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.

For each $i = 2, \dots, n$, solving the linear equation set $H\boldsymbol{c} = \boldsymbol{b}_i$ only needs $O(D\log(D)^2)$ operations if fast polynomial multiplication is used. Summarizing the analyses above, we have the following complexity result for this method.

Theorem 3.2. *Assume that T_1 is constructed (note that $T_2, \dots, T_n$ are not needed). If the minimal polynomial of (6) computed by the Berlekamp–Massey algorithm is of degree D, then the complexity of this method is bounded by*

$$O(D(N_1 + \log(D)) + (n-1)D\log(D)^2) = O(D(N_1 + n\log(D)^2)).$$

This complexity almost matches that of computing the minimal polynomial of the multiplication matrix T_1 if n is small compared with D.

3.1.3. Example

We use the following small example to show how this method applies to ideals in shape position. Given the Gröbner basis of a 0-dimensional ideal $I \subset \mathbb{F}_{11}[x_1, x_2, x_3]$ w.r.t. DRL

$$G_1 = [x_2^2 + 9x_2 + 2x_1 + 6,\ x_1^2 + 2x_2 + 9,\ x_3 + 9],$$

we first compute the degree of I as $D = 4$, the canonical basis $B = [1, x_1, x_2, x_1x_2]$, and the multiplication matrices as

$$T_1 = \begin{pmatrix} 0 & 2 & 0 & 1 \\ 1 & 0 & 0 & 4 \\ 0 & 9 & 0 & 9 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad T_2 = \begin{pmatrix} 0 & 0 & 5 & 7 \\ 0 & 0 & 9 & 5 \\ 1 & 0 & 2 & 4 \\ 0 & 1 & 0 & 2 \end{pmatrix}, \quad T_3 = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}.$$

With the random vector $\boldsymbol{r} = (8, 4, 8, 6)^t \in \mathbb{K}^{(4\times 1)}$, we can construct the linearly recurring sequence

$$s = [8, 4, 0, 7, 6, 8, 10, 10].$$

Then the Berlekamp–Massey algorithm is applied to s to obtain the minimal polynomial $\tilde{f}_1 = x_1^4 + 8x_1 + 9$. From the equality $\deg(\tilde{f}_1) = D = 4$, we know now that the input ideal I is in shape position.

The Hankel coefficient matrix

$$H = \begin{pmatrix} 8 & 4 & 0 & 7 \\ 4 & 0 & 7 & 6 \\ 0 & 7 & 6 & 8 \\ 7 & 6 & 8 & 10 \end{pmatrix}$$

is directly derived from s. Next take the computation of the polynomial $x_2 - f_2(x_1) \in G_2$ for example, the vector $\boldsymbol{b} = (8, 6, 8, 3)^t$ is constructed. The solution of the linear equation set $H\boldsymbol{c} = \boldsymbol{b}$ being $\boldsymbol{c} = (1, 0, 5, 0)^t$, we obtain the polynomial in G_2 as $x_2 + 6x_1^2 + 10$. The other polynomial $x_3 - f_3(x_1)$ can be similarly computed. In the end, we have the Gröbner basis of I w.r.t. LEX

$$G_2 = [x_1^4 + 8x_1 + 9,\ x_2 + 6x_1^2 + 10,\ x_3 + 9].$$

3.2. Deterministic algorithm to compute Gröbner basis of radical of the ideal

As already explained in Remarks 3.1, the classical Wiedemann algorithm is a probabilistic one. For a vector chosen at random, it may only return a proper factor $\tilde{f}_1$ of the polynomial f_1, i.e., $\tilde{f}_1 | f_1$ but $\tilde{f}_1 \neq f_1$. In this case, one may choose to call the Wiedemann algorithm again with another randomly chosen vector to compute another factor $\overline{f}_1$ (possibly still proper) of f_1 and compute $\operatorname{lcm}(\tilde{f}_1, \overline{f}_1)$ in the hope to find f_1. However, this strategy cannot return f_1 in a deterministic way either, and one may expect the lack of efficiency due to the duplicate computation. In fact, the deterministic Wiedemann algorithm can be applied to obtain the univariate polynomial f_1, then one knows for sure whether I is in shape position or not. The main difficulty is to compute the other polynomials $f_2, \ldots, f_n$ in a deterministic way.

In the following we present an algorithm to compute the Gröbner basis of the radical of the ideal I. In particular, a toy example is also presented in Section 3.2.4 to illustrate the way this algorithm works. Indeed, in most applications, only the zeros of a polynomial system are of interest and we do not need to keep their multiplicities. Hence it is also important to design an efficient method to perform the change of ordering of Gröbner basis of an ideal I in a way that the output is the Gröbner basis of $\sqrt{I}$.

3.2.1. Deterministic version of the Wiedemann algorithm

The way how this deterministic variant of the Wiedemann algorithm proceeds is first recalled. Instead of a randomly chosen vector in the classical Wiedemann algorithm, in the deterministic version all the vectors of the canonical basis of $\mathbb{K}^{(D\times 1)}$

$$\boldsymbol{e}_1 = (1, 0, \ldots, 0)^t, \boldsymbol{e}_2 = (0, 1, 0, \ldots, 0)^t, \ldots, \boldsymbol{e}_D = (0, \ldots, 0, 1)^t$$

are used. One first computes the minimal polynomial $f_{1,1}$ of the linearly recurring sequence

$$[\langle \boldsymbol{e}_1, T_1^j \boldsymbol{e}\rangle : j = 0, \ldots, 2D - 1]. \tag{8}$$

Let $d_1 = \deg(f_{1,1})$ and $\boldsymbol{b}_1 = f_{1,1}(T_1)\boldsymbol{e}$. If $\boldsymbol{b}_1 = \boldsymbol{0}$, then one has $f_{1,1} = f_1$ and the algorithm ends; else it is not hard to see that the minimal polynomial $f_{1,2}$ of the sequence

$$[\langle \boldsymbol{e}_2, T_1^j \boldsymbol{b}_1\rangle : j = 0, \ldots, 2(D - d_1) - 1]$$

is indeed a factor of $f_1/f_{1,1}$, a polynomial of degree $\le D - d_1$ (that is why only the first $2(D - d_1)$ terms are enough in the above sequence). Next, one computes $\boldsymbol{b}_2 = f_{1,1} f_{1,2}(T)\boldsymbol{e}$ and checks whether $\boldsymbol{b}_2 = \boldsymbol{0}$. If not, the above procedure is repeated and so on. This method ends with r ($\le D$) rounds and one finds $f_1 = f_{1,1} \cdots f_{1,r}$.

3.2.2. Deterministic algorithm description

First we study the general case when a factor of f_1 is found. Suppose that a vector $\boldsymbol{w} \in \mathbb{K}^{(D\times 1)}$ is chosen to construct the linearly recurring sequence

$$[\langle \boldsymbol{w}, T_1^i \boldsymbol{e}\rangle : i = 0, \ldots, 2D-1], \tag{9}$$

and the minimal polynomial of this sequence is $\tilde{f}_1$, a proper factor of f_1 of degree d. We show how to recover the Gröbner basis of $I + \langle \tilde{f}_1 \rangle$ w.r.t. LEX. Since the ideal I is in shape position, it is not hard to see that the ideal $I + \langle \tilde{f}_1 \rangle$ is also in shape position, and its Gröbner basis w.r.t. LEX is indeed $[\tilde{f}_1, x_2 - \tilde{f}_2, \ldots, x_n - \tilde{f}_n]$, where $\tilde{f}_i$ is the remainder of f_i modulo $\tilde{f}_1$ for $i = 2, \ldots, n$.

Now for each i, we can construct the linear system similar to (7)

$$\langle \boldsymbol{w}, T_1^j T_i \boldsymbol{e}\rangle = \sum_{k=0}^{d-1} y_k \cdot \langle \boldsymbol{w}, T_1^{k+j} \boldsymbol{e}\rangle, \quad j = 0, \ldots, d-1, \tag{10}$$

where $y_0, \ldots, y_{d-1}$ are the unknowns. As the $d \times d$ Hankel matrix of (9) is invertible by Theorem 3.1, there is a unique solution $c_{i,0}, c_{i,1}, \ldots, c_{i,d-1}$ for (10). Next we will connect this solution and a polynomial in the Gröbner basis of $I + \langle \tilde{f}_1 \rangle$, and the following lemma is useful to show this connection.

Lemma 3.3. *Suppose that $\tilde{f}_1$ is the minimal polynomial of (9) for some $\boldsymbol{w} \in \mathbb{K}^{(D\times 1)}$, $\tilde{T}_1$ the multiplication matrix of x_1 of the ideal $I + \langle \tilde{f}_1 \rangle$ w.r.t. $<_1$, and $\tilde{\boldsymbol{e}} = (1, 0, \ldots, 0) \in \mathbb{K}^{(d\times 1)}$ the coordinate vector of $\mathbf{1}$ in $\mathbb{K}[x_1, \ldots, x_n]/(I + \langle \tilde{f}_1 \rangle)$. Then $\tilde{f}_1$ is also the minimal polynomial of $[\tilde{\boldsymbol{e}}, \tilde{T}_1 \tilde{\boldsymbol{e}}, \tilde{T}_1^2 \tilde{\boldsymbol{e}}, \ldots]$.*

Proof. Let $\tilde{f}_1 = x_1^d + \sum_{k=0}^{d-1} a_k x_1^k$. Then according to FGLM criteria, for the ideal $I + \langle \tilde{f}_1 \rangle$,

$$\tilde{T}_1^d \tilde{\boldsymbol{e}} = \sum_{k=0}^{d-1} a_k \tilde{T}_1^k \tilde{\boldsymbol{e}}$$

is the first linear dependency of the vectors $\tilde{\boldsymbol{e}}, \tilde{T}_1 \tilde{\boldsymbol{e}}, \tilde{T}_1^2 \tilde{\boldsymbol{e}}, \ldots$ when one checks the vector sequence $[\tilde{\boldsymbol{e}}, \tilde{T}_1 \tilde{\boldsymbol{e}}, \tilde{T}_1^2 \tilde{\boldsymbol{e}}, \ldots]$. That is to say, $\tilde{f}_1$ is also the minimal polynomial of $[\tilde{\boldsymbol{e}}, \tilde{T}_1 \tilde{\boldsymbol{e}}, \tilde{T}_1^2 \tilde{\boldsymbol{e}}, \ldots]$. □

Proposition 3.4. *Suppose that $\boldsymbol{w} \in \mathbb{K}^{(D\times 1)}$ is such a vector that a proper factor $\tilde{f}_1$ of f_1 of degree $d < D$ is found from the linearly recurring sequence (9). Then for each $i = 2, \ldots, n$, the polynomial $x_i - \sum_{k=0}^{d-1} c_{i,k} x_1^k$, where $c_{i,0}, c_{i,1}, \ldots, c_{i,d-1}$ is the unique solution of (10), is in the Gröbner basis of $I + \langle \tilde{f}_1 \rangle$ w.r.t. LEX.*

Proof. Let $\tilde{T}_1, \ldots, \tilde{T}_d$ be the multiplication matrices of the ideal $I + \langle \tilde{f}_1 \rangle$ w.r.t. $<_1$. We complete the proof in two steps.

Step 1. We connect the polynomial $x_i - \sum_{k=0}^{d-1} c_{i,k} x_1^k$ in the proposition with the target polynomial in the Gröbner basis of $I + \langle \tilde{f}_1 \rangle$ via solutions of two linear equation sets.

For each $i = 2, \ldots, n$, suppose that $x_i - \sum_{k=0}^{d-1} \tilde{c}_{i,k} x_1^k$ is the corresponding polynomial in the Gröbner basis of $I + \langle \tilde{f}_1 \rangle$ w.r.t. LEX. Then $\tilde{T}_i \tilde{\boldsymbol{e}} = \sum_{k=0}^{d-1} \tilde{c}_{i,k} \tilde{T}_1^k \tilde{\boldsymbol{e}}$ holds, and for any vector $\tilde{\boldsymbol{w}} \in \mathbb{K}^{(d\times 1)}$, we have

$$\langle \tilde{\boldsymbol{w}}, \tilde{T}_1^j \tilde{T}_i \tilde{\boldsymbol{e}}\rangle = \sum_{k=0}^{d-1} \tilde{c}_{i,k} \cdot \langle \tilde{\boldsymbol{w}}, \tilde{T}_1^{k+j} \tilde{\boldsymbol{e}}\rangle, \quad j = 0, \ldots, d-1.$$

As long as $\tilde{\boldsymbol{w}}$ is chosen such that the coefficient matrix is invertible, the coefficients $\tilde{c}_{i,0}, \tilde{c}_{i,1}, \ldots, \tilde{c}_{i,d-1}$ will be the unique solution of the linear equation set

$$\langle \tilde{\boldsymbol{w}}, \tilde{T}_1^j \tilde{T}_i \tilde{\boldsymbol{e}}\rangle = \sum_{k=0}^{d-1} y_k \cdot \langle \tilde{\boldsymbol{w}}, \tilde{T}_1^{k+j} \tilde{\boldsymbol{e}}\rangle, \quad j = 0, \ldots, d-1. \tag{11}$$

Therefore, to prove the correctness of the proposition, it suffices to show that there exists $\tilde{\boldsymbol{w}} \in \mathbb{K}^{(d\times 1)}$ such that the coefficient matrix of (11) is invertible, and that the two linear equation sets (10) and (11) share the same solution.

Step 2. To show the correctness of the argument above, we prove a stronger one: the linear equation sets (10) and (11) are the same themselves for some $\tilde{\boldsymbol{w}}$.

To prove that, we need to show that the two Hankel matrices and the vectors in the left hands of (10) and (11) are the same. That is, for some vector $\tilde{\boldsymbol{w}}$

(i) $\langle \boldsymbol{w}, T_1^j \boldsymbol{e}\rangle = \langle \tilde{\boldsymbol{w}}, \tilde{T}_1^j \tilde{\boldsymbol{e}}\rangle$, for $j = 0, \dots, 2d-2$;
(ii) $\langle \boldsymbol{w}, T_1^j T_i \boldsymbol{e}\rangle = \langle \tilde{\boldsymbol{w}}, \tilde{T}_1^j \tilde{T}_i \tilde{\boldsymbol{e}}\rangle$, for $j = 0, \dots, d-1$.

Next we will prove these two arguments respectively.

(i) We take the first d equations in (i)

$$\langle \boldsymbol{w}, T_1^j \boldsymbol{e}\rangle = \langle \tilde{\boldsymbol{w}}, \tilde{T}_1^j \tilde{\boldsymbol{e}}\rangle, \quad j = 0, \dots, d-1.$$

As the vectors $\tilde{\boldsymbol{e}}, \tilde{T}_1 \tilde{\boldsymbol{e}}, \dots, \tilde{T}_1^{d-1}\tilde{\boldsymbol{e}}$ are linearly independent, the above linear equation set has a unique solution $\overline{\boldsymbol{w}}$ for the unknown $\tilde{\boldsymbol{w}}$. From Lemma 3.3, the vector sequence $[\tilde{\boldsymbol{e}}, \tilde{T}_1\tilde{\boldsymbol{e}}, \tilde{T}_1^2\tilde{\boldsymbol{e}}, \dots]$ and the sequence (9) share the same minimal polynomial $\tilde{f}_1$ of degree d. Thus there exist $a_0, \dots, a_{d-1} \in \mathbb{K}$ such that

$$\tilde{T}_1^d \tilde{\boldsymbol{e}} = \sum_{k=0}^{d-1} a_k \tilde{T}_1^k \tilde{\boldsymbol{e}}, \quad \langle \boldsymbol{w}, T_1^d \boldsymbol{e}\rangle = \sum_{k=0}^{d-1} a_k \langle \boldsymbol{w}, T_1^k \boldsymbol{e}\rangle.$$

Hence

$$\langle \overline{\boldsymbol{w}}, \tilde{T}_1^d \tilde{\boldsymbol{e}}\rangle = \langle \overline{\boldsymbol{w}}, \sum_{k=0}^{d-1} a_k \tilde{T}_1^k \tilde{\boldsymbol{e}}\rangle = \sum_{k=0}^{d-1} a_k \langle \overline{\boldsymbol{w}}, \tilde{T}_1^k \tilde{\boldsymbol{e}}\rangle = \sum_{k=0}^{d-1} a_k \langle \boldsymbol{w}, T_1^k \boldsymbol{e}\rangle = \langle \boldsymbol{w}, T_1^d \boldsymbol{e}\rangle.$$

Other equalities in (i) for $j = d+1, \dots, 2d-2$ can also be proved similarly. Actually, the equality $\langle \boldsymbol{w}, T_1^j \boldsymbol{e}\rangle = \langle \overline{\boldsymbol{w}}_0, \tilde{T}_1^j \tilde{\boldsymbol{e}}\rangle$ holds for any $j = 0, 1, \dots$.

(ii) Since there is a polynomial $x_i - \sum_{k=0}^{D-1} a'_k x_1^k$ in the Gröbner basis of I w.r.t. LEX, where $a'_0, \dots, a'_{D-1} \in \mathbb{K}$, we know $T_i \boldsymbol{e} = \sum_{k=0}^{D-1} a'_k T_1^k \boldsymbol{e}$. Then on one hand, for the vector $\boldsymbol{w}$ and any $i = 0, \dots, d-1$, we have

$$\langle \boldsymbol{w}, T_1^j T_i \boldsymbol{e}\rangle = \sum_{k=0}^{D-1} a'_k \langle \boldsymbol{w}, T_1^{k+j} \boldsymbol{e}\rangle.$$

On the other hand, as $x_i - \sum_{k=0}^{D-1} a'_k x_1^k \in I$, we have $x_i - \sum_{k=0}^{D-1} a'_k x_1^k \in I + \langle \tilde{f}_1\rangle$, and thus $\tilde{T}_i \tilde{\boldsymbol{e}} = \sum_{k=0}^{D-1} a'_k \tilde{T}_1^k \tilde{\boldsymbol{e}}$. Therefore for the vector $\overline{\boldsymbol{w}}$ and any $j = 0, \dots, d-1$,

$$\langle \overline{\boldsymbol{w}}, \tilde{T}_1^j \tilde{T}_i \tilde{\boldsymbol{e}}\rangle = \sum_{k=0}^{D-1} a'_k \langle \overline{\boldsymbol{w}}, \tilde{T}_1^{k+j} \tilde{\boldsymbol{e}}\rangle = \sum_{k=0}^{D-1} a'_k \langle \boldsymbol{w}, T_1^{k+j} \boldsymbol{e}\rangle = \langle \boldsymbol{w}, T_1^j T_i \boldsymbol{e}\rangle.$$

This ends the proof. □

Now let us return to the special case of the deterministic Wiedemann algorithm, where unit vectors are used to find $f_1 = f_{1,1} \cdots f_{1,r}$ with $r \le D$ and $\deg(f_{1,i}) = d_i$. Suppose that $\deg(f_1) = D$ so the ideal I is verified in shape position. In the ith step of the algorithm, the unit vector $\boldsymbol{e}_i$ is applied to construct the linearly recurring sequence

$$[\langle \boldsymbol{e}_i, T_1^j \boldsymbol{b}_{i-1}\rangle : j = 0, \dots, 2(D - \prod_{k=1}^{i-1} d_k) - 1],$$

where $\boldsymbol{b}_{i-1} = \prod_{k=1}^{i-1} f_{1,k}(T_1)\boldsymbol{e}$. With this sequence the factor $f_{1,i}$ is computed. As the above sequence is the same as

$$[\langle (\prod_{k=1}^{i-1} f_{1,k}(T_1))^t \boldsymbol{e}_i, T_1^j \boldsymbol{e}\rangle : j = 0, \ldots, 2(D - \prod_{k=1}^{i-1} d_k) - 1],$$

from Proposition 3.4 we can recover efficiently the Gröbner basis of $I + \langle f_{1,i}\rangle$ w.r.t. LEX by constructing and solving linear equation sets with Hankel coefficient matrices.

So we have at hands the factorization $f_1 = f_{1,1} \cdots f_{1,r}$, together with the Gröbner basis of $I + \langle f_{1,i}\rangle$ w.r.t. LEX for $i = 1, \ldots, r$. Suppose that the Gröbner basis for i is

$$P_i = [f_{1,i}, x_2 - f_{2,i}, \ldots, x_n - f_{n,i}]. \tag{12}$$

Then to recover the polynomials f_j in (5) for $j = 2, \ldots, n$, we have the following modulo equation set constructed from $P_1, \ldots, P_r$:

$$\begin{cases} f_j \equiv f_{j,1} \mod f_{1,1}, \\ \quad \vdots \\ f_j \equiv f_{j,r} \mod f_{1,r}. \end{cases} \tag{13}$$

Now it is natural to give a try of the Chinese Remainder Theorem (short as CRT hereafter).

To use the CRT, we have to check first whether $f_{1,1}, \ldots, f_{1,r}$ are pairwise coprime. One simple case is when f_1 is squarefree, or in other words the input ideal I is radical itself. In that case, the direct application of CRT will lead to the Gröbner basis G of I w.r.t. LEX, and the change of ordering ends.

When the polynomial f_1 is not squarefree, the CRT does not apply directly. In this case, the Gröbner basis of $\sqrt{I}$ w.r.t. LEX is our aim. Before the study on how to recover this Gröbner basis, we first make clear how a polynomial set of form (5) can be split to a series of polynomial sets with a certain zero relation according to some factorization of f_1. The following proposition is a direct result of Lazard (1992, Prop. 5(i)), and it is actually a splitting technique commonly used in the theory of triangular sets (Wang, 2001). In what follows, $\mathsf{Z}(F)$ denotes the common zeros of a polynomial set $F \in \mathbb{K}[x_1, \ldots, x_n]$ in $\overline{\mathbb{K}}^n$, where $\overline{\mathbb{K}}$ is the algebraic closure of $\mathbb{K}$.

Proposition 3.5. *Let $T \subset \mathbb{K}[x_1, \ldots, x_n]$ be a polynomial set in the form*

$$[t_1(x_1), x_2 - t_2(x_1), \ldots, x_n - t_n(x_1)],$$

and $t_1 = t_{1,1} \cdots t_{1,r}$. For $i = 1, \ldots, r$, define

$$T(i) = [t_{1,i}, x_2 - t_{2,i}, \ldots, x_n - t_{n,i}],$$

where $t_{j,i}$ is the remainder of t_j modulo $t_{1,i}$ for $j = 2, \ldots, n$. Then we have the following zero relation

$$\mathsf{Z}(T) = \bigcup_{i=1}^{r} \mathsf{Z}(T(i)). \tag{14}$$

Let $\overline{f}_1$ be the squarefree part of f_1. As each P_j in (12) satisfies the form in Proposition 3.5, we can compute t new polynomial sets $\overline{P}_j$ whose univariate polynomials in x_1 is $\overline{f}_{1,j}$ for $j = 1, \ldots, t$, such that $\overline{f}_1 = \prod_{j=1}^{t} \overline{f}_{1,j}$, and $\overline{f}_{1,j}$ are pairwise coprime. These new polynomial sets can be found in the following way. Set $p = \overline{f}_1$. We start with $j = 1$ and compute $\overline{f}_{1,j} = \gcd(f_{1,j}, p)$. As long as this polynomial is not equal to 1, a new polynomial set $\overline{P}_j$ whose univariate polynomial is $\overline{f}_{1,j}$ is constructed from P_j by Proposition 3.5. Next set $p := p/\overline{f}_{1,j}$ and check whether $p = 1$. If so, we know that we already have enough new polynomial sets; otherwise $j := j + 1$, and the process above is repeated.

Now we reduce the current case to the earlier one with $\overline{f}_1$ squarefree and $\overline{P}_1, \ldots, \overline{P}_t$ to construct the modulo equation sets. Thus the Gröbner basis of $\sqrt{I}$ w.r.t. LEX can be obtained similarly (note that extracting the squarefree part of f_1 results in the radical of I). We would like to remark that when we have to compute $\overline{P}_j$ as above, it means that the input ideal I itself is not radical and vice versa. So the corresponding probability to take this step is equal to that of the input ideal I being in shape position but not radical.

The whole method based on the deterministic Wiedemann algorithm is summarized in Algorithm 4 below. The subfunction Sqrfree() returns the squarefree part of the input polynomial. The operator "cat" means concatenating two sequences.

Algorithm 4: Shape position (deterministic) $G_2 :=$ ShapePositionDeterministic($G_1, <_1$).

```
Input: $G_1$, Gröbner basis of a 0-dimensional ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ w.r.t. $<_1$
Output: $G_2$, Gröbner basis of $\sqrt{I}$ w.r.t. LEX if $I$ is in shape position; Fail, otherwise.
1  Compute the canonical basis of $\mathbb{K}[x_1, \dots, x_n]/\langle G_1 \rangle$ and multiplication matrices $T_1, \dots, T_n$;
2  $\boldsymbol{e}_1 := (1, 0, \dots, 0)^t, \boldsymbol{e}_2 := (0, 1, 0, \dots, 0)^t, \dots, \boldsymbol{e}_D := (0, \dots, 0, 1)^t \in \mathbb{K}^{(D\times 1)}$;
3  $k := 1$;  $F := []$;  $f; = 1$;  $d := 0$;  $\boldsymbol{b} = \boldsymbol{e}_1$;  $S = []$;
4  while $\boldsymbol{b} \neq \boldsymbol{0}$ do
5      $s := [\langle \boldsymbol{e}_k, T_1^i \boldsymbol{b} \rangle : i = 0, 1, \dots, 2(n-d) - 1]$;
6      $g :=$ BerlekampMassey($s$);
7      $f := f \cdot g$;  $d := \deg(f)$;  $F := F$ cat $[g]$;  $\boldsymbol{b} := g(T_1)\boldsymbol{b}$;  $S := S$ cat $[s]$;
8      $k := k + 1$;
9  end
10 (Suppose that $F = [f_{1,1}, \dots, f_{1,r}]$) $f_1 := \prod_{i=1}^r f_{1,i}$;
11 if $\deg(f_1) \neq D$ then
12     return Fail
13 else
14     for $i = 1, \dots, r$ do
15         $d_i := \deg(f_{1,i})$;
16         for $j = 2, \dots, n$ do
17             Construct the Hankel matrix $H_j$ and the vector $\boldsymbol{b}$ from $S$;
18             Compute $\boldsymbol{c} = (c_1, \dots, c_{d_i})^t := H_j^{-1}\boldsymbol{b}$;  $f_{i,j} := \sum_{k=0}^{d_i} \boldsymbol{c}_{k+1} x_1^k$;
19         end
20     end
21     $\overline{f}_1 :=$ Sqrfree($f_1$);
22     if $\overline{f}_1 \neq f_1$ then
23         Compute $\{[\overline{f}_{1,j}, x_2 - \overline{f}_{2,j}, \dots, x_n - \overline{f}_{n,j}] : j = 1, \dots, t\}$ from $\{[f_{1,i}, x_2 - f_{2,i}, \dots, x_n - f_{n,i}] : i = 1, \dots, r\}$ by
           Proposition 3.5 such that $\overline{f}_1 = \prod_{j=1}^t \overline{f}_{1,j}$ and $\overline{f}_{1,j}$ are pairwise coprime;
24     end
25     for $j = 2, \dots, n$ do
26         Solve the modulo equation set (13) to get $f_j$;
27     end
28     return $[\overline{f}_1, x_2 - f_2, \dots, x_n - f_n]$
29 end
```

Remark 3.2. If the factors $f_{1,1}, \dots, f_{1,r}$ of f_1 returned by the deterministic Wiedemann algorithm are pairwise coprime (which needs extra computation to test), the Gröbner basis of I w.r.t. LEX can be computed from the CRT.

The method of the deterministic version described above is also applicable to the Wiedemann algorithm with several random vectors. To be precise, when the first random vector does not return the correct polynomial f_1, one may perform a similar procedure as the deterministic Wiedemann algorithm by updating the sequence with a newly chosen random vector (instead of $\boldsymbol{e}_i$ in the basis) and repeating (Wiedemann, 1986). In that case, the method above with CRT can also be used to compute the Gröbner basis of $\sqrt{I}$ w.r.t. LEX.

3.2.3. Complexity

Next the computational complexity, namely the number of field operations needed, for the deterministic method for ideals in shape position is analyzed.

(a) In total the deterministic Wiedemann algorithm needs

$$O(D(N_1 + D\log(D)\log\log(D)))$$

operations if fast polynomial multiplications are used (Wiedemann, 1986). Here N_1 still denotes the number of nonzero entries in T_1.

(b) Next at most D structured linear equation sets with Hankel coefficient matrices are constructed and solved, each with maximum operations $O(D\log(D)^2)$. Hence this procedure needs $O(D^2\log(D)^2)$ operations at most.

(c) With fast multiplications, the squarefree part $\overline{f}_1$ of f_1 can be obtained with complexity at most $O(D\log(D)^2)$ for the case when $\mathbb{K}$ has characteristic 0 and $O(D\log(D)^2 + D\log(q/p))$ for characteristic $p > 0$ respectively, where $|\mathbb{K}| = q$ (Von zur Gathen and Gerhard, 2003, Thm. 14.20 and Exer. 14.30). For the case when f_1 is not squarefree, suppose that t new polynomial sets $\overline{P}_1, \ldots, \overline{P}_t$ are needed, and $\deg(\overline{f}_{1,i}) = d_i$ for $i = 1, \ldots, t$. To compute each set $\overline{P}_i$ of the form (5), $n-1$ polynomial divisions are needed to find the remainders, with complexity $O(nd_iD)$. Hence the total complexity to obtain $\overline{P}_1, \ldots, \overline{P}_t$ is

$$O(\sum_{i=1}^{t} nd_iD) = O(nD\sum_{i=1}^{t} d_i) \le O(nD^2),$$

for we have $\sum_{i=1}^{t} d_i = \deg(\overline{f}_1) < D$.

(d) Solving the modulo equation set (13) for each $j = 2, \ldots, n$ requires $O(D\log(D)^2)$ operations at most by Von zur Gathen and Gerhard (2003, Thm. 5.7). Thus in total $O(nD\log(D)^2)$ operations are needed for the CRT application.

Therefore, we have the following complexity result for the method with the deterministic Wiedemann algorithm.

Theorem 3.6. *Assume that T_1 is known. If the input ideal I is in shape position, then this deterministic method will return the Gröbner basis of $\sqrt{I}$ w.r.t. LEX with the complexity*

$$O(D(N_1 + D\log(D)^2 + nD + R)),$$

where $R = 0$ if $\mathbb{K}$ has characteristic 0 and $R = \log(q/p)$ if $\mathbb{K}$ has characteristic $p > 0$ and $|\mathbb{K}| = q$.

3.2.4. Example

Here is a toy example to illustrate how the deterministic method works. Consider an ideal I in $\mathbb{F}_2[x_1, x_2]$ generated by its Gröbner basis w.r.t. DRL

$$G_1 := [x_2x_1^3 + x_1^3 + x_1 + 1, x_1^4 + x_1^3 + x_2 + 1, x_1^2 + x_2^2].$$

Its Gröbner basis w.r.t. LEX is

$$G_2 = [f_1 := (x_1+1)^3(x_1^2 + x_1 + 1)^2, x_2 + x_1^4 + x_1^3 + 1],$$

from which one can see that I is in shape position.

From G_1 the canonical basis $B = [1, x_1, x_2, x_1^2, x_1x_2, x_1^3, x_1^2x_2]$ and the multiplication matrices T_1 and T_2 are computed. With a vector $\boldsymbol{r} = (1, 1, 0, 1, 0, 1, 0)^t \in \mathbb{F}_2^{(7\times 1)}$ generated at random, the classical Wiedemann algorithm will only return a proper factor $(x_1+1)(x_1^2 + x_1 + 1)$ of f_1, and whether I is in shape position is unknown.

Next we use the deterministic Wiedemann algorithm to recover f_1. With $\boldsymbol{e}_1 = (1, 0, \ldots, 0)^t$, a factor $f_{1,1} = (x_1+1)^2(x_1^2+x_1+1)$ of f_1 is found with the Berlekamp–Massey applied to the sequence (8). Then we update the vector

$$\boldsymbol{b} = f_{1,1}(T_1)\boldsymbol{e} = (0, 1, 1, 0, 0, 0, 0)^t,$$

and execute the second round with $\boldsymbol{e}_2 = (0, 1, 0, \ldots, 0)^t$, obtaining another factor $f_{1,2} = (x_1+1)(x_1^2 + x_1 + 1)$. This time the updated vector $\boldsymbol{b} = \boldsymbol{0}$, and thus the deterministic Wiedemann algorithm ends, and f_1 is computed as $f_{1,1}f_{1,2}$. As $\deg(f_{1,1}f_{1,2}) = D$, now I is verified to be in shape position.

Then we construct the linear equation sets similar to (7) to recover $f_{2,1}$ and $f_{2,2}$ respectively. The first one, for example, is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

After solving them, we have the Gröbner bases of $I + \langle f_{1,1} \rangle$ and $I + \langle f_{1,2} \rangle$ respectively as

$$P_1 = [(x_1+1)^2(x_1^2+x_1+1), x_2+x_1],$$
$$P_2 = [(x_1+1)(x_1^2+x_1+1), x_2+x_1].$$

Then the squarefree part $\overline{f}_1$ of f_1 is computed, and we find that I is not radical, and thus only the Gröbner basis $\tilde{G}_2$ of $\sqrt{I}$ w.r.t. LEX may be computed. From $f_{1,2} = \overline{f}_1$, we directly have $\tilde{G}_2 = P_2$, and the algorithm ends.

The way to compute $\tilde{G}_2$ by CRT, which is more general, is also shown in the following. Two new polynomial sets

$$\overline{P}_1 = [x_1+1, x_2+1], \quad \overline{P}_2 = [x_1^2+x_1+1, x_2+x_1]$$

may be first computed and selected according to $\overline{f}_1$ by Proposition 3.5. Then the modulo equation set

$$\begin{cases} f_2 \equiv 1 & \text{mod } x_1+1, \\ f_2 \equiv x_1 & \text{mod } x_1^2+x_1+1 \end{cases}$$

as (13) is solved with CRT, resulting in the same $\tilde{G}_2$. One can check that $\tilde{G}_2$ is the Gröbner basis of $\sqrt{I}$ w.r.t. LEX with any computer algebra system.

3.3. *Incremental algorithm to compute the univariate polynomial*

For a 0-dimensional ideal $I \subset \mathbb{K}[x_1, \ldots, x_n]$, the univariate polynomial in its Gröbner basis w.r.t. LEX is of special importance. For instance, it may be the only polynomial needed to solve some practical problems. Furthermore, in the case when $\mathbb{K}$ is a finite field, after the univariate polynomial is obtained, it will not be hard to compute all its roots, then one can simplify the original polynomial system by substituting the roots back, and sometimes the new system will become quite easy to solve.

Besides the two methods in the previous parts, next the well-known incremental Wiedemann algorithm dedicated to computation of the univariate polynomial is briefly recalled and discussed.

In the Wiedemann algorithm, the dominant part of its complexity comes from construction of the linearly recurring sequence ($O(DN_1)$), while the complexity of the Berlekamp–Massey algorithm is relatively low ($O(D\log(D))$ for fast Berlekamp–Massey algorithm). Hence the idea of the incremental method is to construct the sequence incrementally to save computation and apply the Berlekamp–Massey algorithm to each incremental step.

We start with the linearly recurring sequence $[\langle \boldsymbol{r}, T_1^i \boldsymbol{e} \rangle : i = 0, 1]$ and compute its minimal polynomial with the Berlekamp–Massey algorithm. Next we proceed step by step with the sequence

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

until the returned polynomial coincides with the one in the previous step. Then this minimal polynomial equals the univariate polynomial f we want to compute with a large probability.

Suppose that $\deg(f) = d$. Then the number of steps the method takes is bounded by $d+1$. In other words, the method stops at most after the sequence $[\langle \boldsymbol{r}, T_1^i \boldsymbol{e} \rangle : i = 0, \ldots, 2d+1]$ is handled. The number of field operations to construct the sequences is $O(dN_1)$, while the total complexity to compute the minimal polynomials with the Berlekamp–Massey algorithm is $O(\sum_{k=1}^{d+1} k^2) = O(d^3)$. Therefore the overall complexity for the incremental Wiedemann method to compute the univariate polynomial is $O(dN_1 + d^3)$. As can be seen here from this complexity, the incremental method is sensitive to the output polynomial f. When the degree d is relatively small compared with D, this method will be useful.

Remark 3.3. One may use extra techniques to further reduce the total complexity $O(dN_1 + d^3)$ of this incremental algorithm. For example, with the fast Berlekamp–Massey algorithm with complexity $O(k\log(k))$, the total complexity can be reduced to $O(dN_1 + d^2\log(d))$, and this can be further reduced to $O(dN_1 + d\log(d))$ if we only apply the Berlekamp–Massey algorithm when the length of the sequence grows to a power of 2. However, we would like to mention that usually the degree d is small compared with N_1, and thus in this case the dominant part in the total complexity is dN_1.

4. General case: BMS-based algorithm

In the general case when the ideal I may not be in shape position, those methods described in Section 3 may not be applicable. However, we still want to follow the idea of constructing linearly recurring sequences and computing their minimal polynomials with the Berlekamp–Massey algorithm. The way to do so is to generalize the linearly recurring sequence to a multi-dimensional linearly recurring relation and apply the BMS algorithm to find its minimal generating set.

4.1. Description of BMS-based algorithm

Given the multiplication matrices $T_1, \ldots, T_n$, we first define an n-dimensional mapping $E : \mathbb{Z}^n_{\geq 0} \longrightarrow \mathbb{K}$ as

$$(s_1, \ldots, s_n) \longmapsto \langle \boldsymbol{r}, T_1^{s_1} \cdots T_n^{s_n} \boldsymbol{e} \rangle, \tag{15}$$

where $\boldsymbol{r} \in \mathbb{K}^{(D\times 1)}$ is a random vector. One can see that such a mapping is an n-dimensional generalization of the linearly recurring sequence constructed in the Wiedemann algorithm. Note that $T_1^{s_1} \cdots T_n^{s_n}\boldsymbol{e}$ in the definition of E above is the coordinate vector of $(s_1, \ldots, s_n)$ in the FGLM algorithm. As a polynomial f in the Gröbner basis of I is of form (2) and the linear dependency (1) holds, one can verify that f satisfies (3) and thus is a polynomial in $I(E)$.

Next we apply the BMS algorithm to compute the Gröbner basis of $I(E)$, in the hope that the computed Gröbner basis is equal to that for I. Then as polynomials in the Gröbner basis of I are characterized by the linear dependency in (1), we are able to check whether the equality between the Gröbner basis computed by the BMS algorithm and that of I we want to compute.

We remark that f is in $I(E)$ for any vector $\boldsymbol{r}$. The method we propose above is indeed a multi-dimensional generalization of the Wiedemann algorithm. Recall that the minimal polynomial g of the Krylov sequence $[\boldsymbol{b}, A\boldsymbol{b}, A^2\boldsymbol{b}, \ldots]$ is what the Wiedemann algorithm seeks, for g directly leads to a solution of the linear equation $A\boldsymbol{x} = \boldsymbol{b}$ for a nonsingular matrix A and vector $\boldsymbol{b}$. Then a random vector is chosen to convert the sequence to a scalar one

$$[\langle \boldsymbol{r}, \boldsymbol{b}\rangle, \langle \boldsymbol{r}, A\boldsymbol{b}\rangle, \langle \boldsymbol{r}, A^2\boldsymbol{b}\rangle, \ldots],$$

and the Berlekamp–Massey algorithm is applied to find the minimal polynomial of this new sequence, in the hope that g can be obtained. While the method proposed here converts the mapping from $(s_1, \ldots, s_n)$ to its coordinate vector in the FGLM to an n-dimensional scalar mapping with a random vector, and then the BMS algorithm (generalization of Berlekamp–Massey) is applied to find the minimal polynomial set, which is also the Gröbner basis, w.r.t. to a term ordering.

This method for computing the Gröbner basis of I makes full use of the sparsity of $T_1, \ldots, T_n$ in the same way as how the Wiedemann algorithm takes advantage of the sparsity of A. The method is a probabilistic one, also the same as the Wiedemann algorithm. This is reasonable for the ideal $I(E)$ derived from the n-dimensional mapping may lose information of I because of the random vector, with $I \subset I(E)$. Clearly, when I is maximal (corresponding to the case when g in the Wiedemann algorithm is irreducible), $I(E)$ will be equal to I.

With all the discussions, the algorithm is formulated as follows. The “Termination Criterion” means the termination criterion for the BMS algorithm to end, as discussed in the following Section 4.2 for the LEX ordering. The subfunction Reduce(F) performs reduction on F so that every polynomial $f \in F$ is reduced w.r.t. $F \setminus \{f\}$, and this subfunction is only called when the target term ordering is LEX (also discussed in Section 4.2 below). The subfunction IsGB(F) is to check whether F is the Gröbner basis of I w.r.t. $<_2$ by checking whether each polynomial in F satisfies the condition (1) and the degree of $\langle F\rangle$ is equal to D, and it returns true if F is the Gröbner basis of I w.r.t. $<_2$ and false otherwise.

Algorithm 5: General case $G_2 :=$ GeneralBMSbased(G_1, $<_1$).

```
Input: $G_1$, Gröbner basis of a 0-dimensional ideal $I \subset \mathbb{K}[x_1,\ldots,x_n]$ w.r.t. $<_1$
Output: Gröbner basis of $I$ w.r.t. $<_2$; or Fail, if the BMS algorithm fails returning the correct Gröbner basis
1  Compute the canonical basis of $\mathbb{K}[x_1,\ldots,x_n]/\langle G_1\rangle$ and multiplication matrices $T_1,\ldots,T_n$;
2  Choose $\boldsymbol{r} \in \mathbb{K}^{(D\times 1)}$ at random;
3  $\boldsymbol{u} := \boldsymbol{0}$;  $F := [1]$;  $G := [\ ]$;  $E := [\ ]$;
4  repeat
5      $e := \langle \boldsymbol{r}, T_1^{\boldsymbol{u}_1}\cdots T_n^{\boldsymbol{u}_n}\boldsymbol{e}\rangle$;
6      $E := E$ cat $[e]$;
7      $F, G :=$ BMSUpdate($F, G, \boldsymbol{u}, E$) ;
8      $\boldsymbol{u} :=$ Next($\boldsymbol{u}$) w.r.t. $<_2$;
9      if $<_2 = LEX$ then
10         $F :=$ Reduce($F$);
11     end
12 until Termination Criterion;
13 if IsGB($F$) then
14     return $F$;
15 else
16     return Fail;
17 end
```

The correctness of Algorithm 5 is obvious because of Lines 13–17. The termination of this algorithm is studied in Mou (2012), and we briefly outline the proof here. Clearly it suffices to prove the termination of the loop inside. In fact, when the polynomial set F the BMS algorithm maintains turns to the Gröbner basis of $I(E)$ w.r.t. $<_2$, the designed termination criterion, which is mainly based on that F keeps unchanged for a certain number of passes, will be satisfied. And a sufficient condition for F being the Gröbner basis is given as Theorem 4.4 below.

4.2. BMS algorithm for LEX

Graded term orderings are usually used in the BMS algorithm in literature due to their applications in Coding Theory (Cox et al., 1998), and the BMS algorithm works as designed for graded term orderings like DRL. However, as explained before, the LEX ordering is of our main interest, and there arise problems for the BMS algorithm applied to the LEX ordering. Hence we have modified the original BMS algorithm for the particular LEX ordering and these modifications are presented in the following.

Remark 4.1. When the term ordering in the BMS algorithm is LEX, computation of the univariate polynomial in this algorithm is exactly the same as that 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.

In the BMS algorithm, after each call of BMSUpdate() at some term $\boldsymbol{u}$, the polynomial set F and the delta set in the algorithm are updated, and the set of leading terms of updated polynomials in F and the updated delta set determine each other. In our modified BMS algorithm for the LEX ordering, an extra reduction on F is performed after every call of BMSUpdate() to control the size of intermediate polynomials. This is not necessary in the case of graded orderings like DRL, for the number of terms in each polynomial in F is bounded by the number of terms smaller than its leading term, and thus the sizes of polynomials in F are controlled in this way. However, for the LEX ordering, the leading terms will not bound the number of terms in polynomials in F. For example, after the call of BMSUpdate(), the polynomial set F could be $\{x^{10}+3x^8+2x^2+5, x^5y+x^{100}+\cdots\}$ for the LEX ordering with $x<y$, and without the reduction on F there may be an intermediate expression swell.

As explained in Section 2.2, the BMS algorithm will call BMSUpdate() term by term following the term ordering until it reaches the termination term. Hence for graded term orderings like DRL, once the termination term is fixed, one can predict precisely all the terms which will be handled in the BMS algorithm. However, for the LEX ordering it is not the case.

Take an ideal $I(E)\subset\mathbb{K}[x_1,x_2]$ for example (one may also refer to Fig. 1 for an illustration). When the term ordering is LEX, the BMS algorithm first handles the terms $[1,x_1,x_1^2,\ldots]$ and it degenerates to the Berlekamp–Massey as indicated in Remark 4.1. Then the BMS algorithm moves on to handle the terms $[x_2,x_1x_2,x_1^2x_2,\ldots]$. However, without any modification, the original BMS algorithm will continue to handle terms in this infinite sequence endlessly. One possible solution to this problem is to assign manually a bound l_1 large enough on the degree of the variable x_1 and end the computation of BMS algorithm along this infinite sequence if the term it handles reaches $x_1^{l_1}x_2$. By assigning another similar bound l_2 on x_2, one may fix the terms for the BMS algorithm to handle as $\{x_1^ix_2^j : 1\le i\le l_1, 1\le j\le l_2\}$. However, this strategy may involve a lot of useless terms where the polynomial set F in the BMS algorithm does not change at all. This drawback could be illustrated as in Fig. 1: if one assign the bounds $l_1=29$ (indicated by the Berlekamp–Massey algorithm) and $l_2=20$ (some integer large enough), then the computation for BMS to handle the sequence of terms

$$[x_1^{15}x_2^4, x_1^{16}x_2^4,\ldots,x_1^{29}x_2^4]$$

will be useless because the polynomial set F does not change (in Fig. 1, the terms where F change are marked as diamonds). Based on the analysis above, for the BMS algorithm for the LEX ordering, we need a new termination criterion with each variable taken into consideration.

In fact, the termination criterion for the BMS algorithm is a key problem in its study. There exist theoretical results on the termination of the algorithm on the assumption of knowing the resulting delta set (see Theorem 4.4 below for details), but this is not useful in practice, for one can never know the output before the algorithm ends. In principle, the termination criterion we use is based on the fact that if the polynomial sets maintained by the BMS algorithm keep unchanged for a number of terms, then the probability for the algorithm to end is large when the ground field $\mathbb{K}$ is of large cardinality. Certainly this termination criterion makes the BMS algorithm probabilistic, but as shown in Algorithm 5, our BMS-based algorithm for change of ordering of Gröbner bases is a Las Vegas algorithm, and this kind of termination criterion does not change this essence. In particular, as indicated in previous analyses, we also add a termination check (based on the same criterion that the polynomial sets keep unchanged) for each variable x_i. The specific termination criterion for the BMS algorithm for LEX is presented in Appendix A, and the readers may refer to Mou (2012) for more details.

4.3. Complexity

We focus on the case when the target term ordering is LEX. Following the previously made assumption that all the multiplication matrices are known, one sees that there is no arithmetic needed in Lines 1–3 of Algorithm 5.

To simplify the computation at Line 5, we record the coordinate vector in the inner product each time this line is executed. Suppose that the coordinate vector

$$\tilde{\boldsymbol{e}} = T_1^{\boldsymbol{u}_1}\cdots T_i^{\boldsymbol{u}_i-1}\cdots T_n^{\boldsymbol{u}_n}\boldsymbol{e}$$

for a certain term $(u_1,u_2,\ldots,u_{i-1},u_i-1,u_{i+1},\ldots,u_n)$ has been computed and recorded. Then we know the value at $\boldsymbol{u}=(u_1,\ldots,u_n)$ is

$$\langle\boldsymbol{r}, T_1^{\boldsymbol{u}_1}\cdots T_n^{\boldsymbol{u}_n}\boldsymbol{e}\rangle = \langle\boldsymbol{r}, T_i\tilde{\boldsymbol{e}}\rangle,$$

for all T_i and T_j commute. Thus the computation of one value of E at Line 5 can be achieved within $O(N)$ operations, where N is the maximal number of nonzero entries in matrices $T_1,\ldots,T_n$.

For Line 7, the complexities of the three steps in Algorithm 2 (BMSUpdate()) are analyzed below.

(a) As an extra reduction step is applied after each iteration, the numbers of terms of polynomials in F are bounded by $D+1$. Denote by $\hat{N}$ the number of polynomials in G_2. Then checking whether F is valid up to $\text{Next}(\boldsymbol{u})$ needs $O(\hat{N}D)$ operations.

(b) The computation of the new delta set $\Delta(\text{Next}(\boldsymbol{u}))$ only involves integer computations, and thus no field operation is needed.

(c) Constructing the new polynomial set F^+ valid up to $\text{Next}(\boldsymbol{u})$ requires $O(\hat{N}D)$ operations at most. The readers may refer to Saints and Heegard (2002), Cox et al. (1998) for the way to construct new polynomials.

After the update is complete, a polynomial reduction is applied to F to control the size of every polynomial at Line 10. This requires $O(\hat{N}\overline{N}D)$ operations, where $\overline{N}$ 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 5 is

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

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.

Theorem 4.1. *Suppose that the input ideal $I\subset\mathbb{K}[x_1,\ldots,x_n]$ is of degree D. Then the number of passes of the loop in Algorithm 5 is bounded by $2nD$.*

Before giving the proof, we need to introduce some of the proven results on the BMS algorithm for preparations. Refer to Bras-Amorós and O'Sullivan (2006), Cox et al. (1998) for more details.

Denote the previous term of $\boldsymbol{u}$ w.r.t. $<$ by $\text{Pre}(\boldsymbol{u})$. Given an n-dimensional array E, suppose that a polynomial $f\in\mathbb{K}[x_1,\ldots,x_n]$ is valid for E up to $\text{Pre}(\boldsymbol{u})$ but not to $\boldsymbol{u}$. Then the term $\boldsymbol{u}-\text{lt}(f)$ is called the *span* of f and denoted by $\text{Span}(f)$, while the term $\boldsymbol{u}$ is called the *fail* of f and written as $\text{Fail}(f)$. When $f\in I(E)$, f is valid up to every term, and in this case we define $\text{Span}(f):=\infty$. The following proposition reveals the importance of spans.

Proposition 4.2 *(Bras-Amorós and O'Sullivan, 2006, Cor. 9). $\Delta(E)=\{\text{Span}(f): f\notin I(E)\}$.*

Define $I(\boldsymbol{u})=\{f\in\mathbb{K}[x_1,\ldots,x_n]:\text{Fail}(f)>\boldsymbol{u}\}$. Such a set is not an ideal but is closed under monomial multiplication: supposing that $F\in I(\boldsymbol{u})$, we have $\boldsymbol{t}F\in I(\boldsymbol{u})$ for every term $\boldsymbol{t}\in\mathbb{K}[x_1,\ldots,x_n]$.

Proposition 4.3 *(Bras-Amorós and O'Sullivan, 2006, Prop. 6). For each $\boldsymbol{u}$, $\Delta(\boldsymbol{u})=\{\text{Span}(f): f\notin I(\boldsymbol{u})\}$. Furthermore, $\boldsymbol{v}\in\Delta(\boldsymbol{u})\setminus\Delta(\text{Pre}(\boldsymbol{u}))$ if and only if $\boldsymbol{v}\prec\boldsymbol{u}$ and $\boldsymbol{u}-\boldsymbol{v}\in\Delta(\boldsymbol{u})\setminus\Delta(\text{Pre}(\boldsymbol{u}))$.*

The above proposition states when a term in $\Delta(E)$ is determined, and it will be used extensively in the sequel. Also from this proposition, one can derive the following termination criterion for the BMS algorithm, which are mainly designed for graded term orderings like DRL.

Theorem 4.4 *(Cox et al., 1998, pp. 529, Prop. (3.12)). Let c_{max} be the largest element of $\Delta(E)$ and s_{max} be the largest element of $\{\text{lt}(g): g\in G\}$, where G is the Gröbner basis of $I(E)$ w.r.t. $<$.*

(a) *For all $\boldsymbol{u}\geq c_{max}+c_{max}$, $\Delta(\boldsymbol{u})=\Delta(E)$ holds.*
(b) *For all $\boldsymbol{v}\geq c_{max}+\max\{c_{max},s_{max}\}$, the polynomial set F the BMS algorithm maintains equals G.*

Next we first illustrate the procedure for a 2-dimensional example derived from Cyclic5, where the term ordering is LEX with $x_1<x_2$. Both the resulting delta set (marked with crosses) and all the terms handled by the BMS algorithm (with diamonds) are recorded in Fig. 1. As the term ordering is LEX, in this figure the diamond terms are handled by the BMS algorithm row by row and bottom up, while in each row from left to right.

Now we go into some details of what happens when a diamond row is handled by the BMS algorithm. We call a diamond (or cross) row the jth diamond (or cross) row if terms in this row are

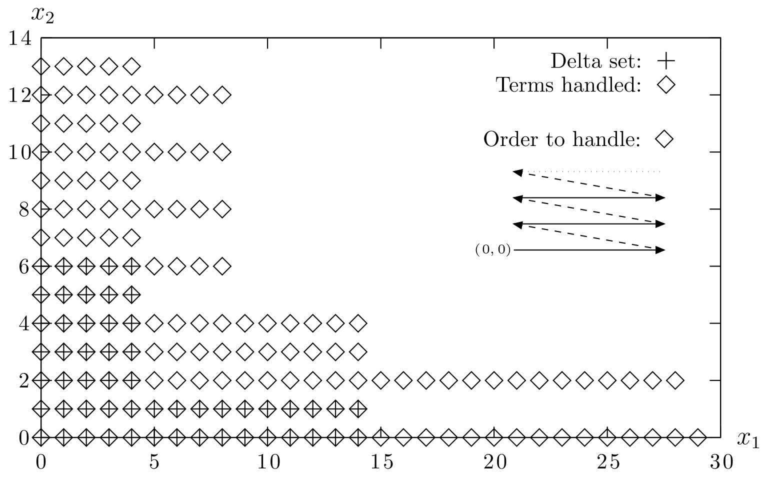

Fig. 1. Delta set (+) and terms needed (⋄) for Cyclic5-2.

(i, j). Then for the 0th diamond row, the BMS algorithm degenerates to the Berlekamp–Massey one to compute the univariate polynomial $f_1(x_1)$. Here 30 diamond terms are needed because the minimal polynomial is of degree 15.

(a) For the jth diamond row with an odd j in Fig. 1, from Proposition 4.3 one knows that when the terms in it are handled by the BMS algorithm, the delta set does not change (one may refer to Table 1 to find the same behavior). Such diamond rows are only bounded by the latest verified row in the delta set. For example, the 3rd diamond row is of the same length as the 1st cross row, while the 5th diamond row is as that of the 2nd cross one.

This is because for a latest verified cross row, there is a term which is the leading term of some polynomial in the target Gröbner basis associated to it (e.g., the term $(15, 0)$ associated to the 2nd cross row in Fig. 1). According to the FGLM criteria, this means that there is a linear dependency of the vectors $T_1^i T_2^j \boldsymbol{e}$ and thus that of the corresponding values $\langle \boldsymbol{r}, T_1^i T_2^j \boldsymbol{e}\rangle$. Furthermore, since along such a jth diamond row the delta set does not change, the leading terms of polynomials in F do not change either. Hence, by the definition of polynomials valid up to some term (Equation (4)), one knows that if F is valid up to the largest term, say $(\tilde{i}, j)$, in such a jth diamond row bounded by the latest verified cross row, then F is also valid up to any term $(\tilde{i} + k, j)$ in the diamond row for $k = 1, 2, \ldots$.

(b) For a 2kth diamond row, its number is related to two criteria. On one hand, again from Proposition 4.3, the kth cross row is determined while the 2kth diamond row is handled in the BMS algorithm. Denote by $c_{max}(k)$ the largest term in the kth cross row, then terms up to $c_{max}(k) + c_{max}(k)$ in the 2kth diamond row have to be handled to furnish the kth cross row. On the other hand, the number of 2kth diamond row is also bounded by the latest verified cross row, as the odd diamond ones. The first criterion is shown by the 6th diamond and the 3rd cross rows, while the 4th diamond row is the result of both criteria.

The c_{max} and s_{max} in Theorem 4.4 are respectively $(4, 6)$ and $(0, 7)$. In fact, the BMS algorithm obtains the whole delta set at $(8, 12) = c_{max} + c_{max}$, and the polynomial set it maintains grows to the Gröbner basis at $(4, 13) = c_{max} + s_{max}$, which is also where the algorithm ends.

For a term $\boldsymbol{u} = (u_1, \ldots, u_i, 0, \ldots, 0) \in \mathbb{K}[x_1, \ldots, x_i]$, in the proof below we write it as $\boldsymbol{u} = (u_1, \ldots, u_i)$ for simplicity, ignoring the last $n - i$ zero components in the terms.

Proof of Theorem 4.1. Suppose that G is the Gröbner basis of $I(E)$ the BMS algorithm computes. Denote the number of terms needed in the BMS algorithm to compute $G \cap \mathbb{K}[x_1, \ldots, x_i]$ by χ_i, and $\Delta_i := \Delta(E) \cap \mathbb{K}[x_1, \ldots, x_i]$. From $I \subseteq I(E)$ one knows that $\Delta(E)$ is a subset of the canonical basis of

$\mathbb{K}[x_1,\ldots,x_n]/I$, thus $|\Delta(E)| \le D$. Therefore to prove the theorem, it suffices to show that $2n|\Delta(E)|$ is an upper bound.

We make induction on the number of variable i of $\mathbb{K}[x_1,\ldots,x_i]$. For $i=1$, the BMS algorithm degenerates to the Berlekamp–Massey, and one can easily see that $\chi_1 \le 2|\Delta_1|$ holds. Now suppose $\chi_k \le 2k|\Delta_k|$ for $k(<n)$. Next we prove $\chi_{k+1} \le 2(k+1)|\Delta_{k+1}|$.

As previously explained, in the terms to compute $G \cap \mathbb{K}[x_1,\ldots,x_{k+1}]$, the terms $(u_1,\ldots,u_k,2l)$ are determined by two factors: terms $(v_1,\ldots,v_k,l)$ in Δ_{k+1}, and the latest verified terms in the delta set. First we ignore those $(u_1,\ldots,u_k,2l)$ terms determined by the latter criterion, and denote by $\mathcal{T}_{k+1}$ all the remaining ones in $\mathbb{K}[x_1,\ldots,x_{k+1}]$. We claim that $|\mathcal{T}_{k+1}|$ is bounded by $(2k+1)|\Delta_{k+1}|$.

From Theorem 4.4, we can suppose there exists some integer m, such that

$$\mathcal{T}_{k+1} = \bigcup_{l=0,\ldots,2m+1} \mathcal{T}_{k+1,l}, \quad \Delta_{k+1} = \bigcup_{j=0,\ldots,m} \Delta_{k+1,j}, \tag{16}$$

where

$$\mathcal{T}_{k+1,l} := \{\boldsymbol{u} \in \mathcal{T}_{k+1} : \boldsymbol{u} = (u_1,\ldots,u_k,l)\},$$
$$\Delta_{k+1,j} := \{\boldsymbol{u} \in \Delta_{k+1} : \boldsymbol{u} = (u_1,\ldots,u_k,j)\}.$$

Clearly $|\mathcal{T}_{k+1,0}| = \chi_k \le 2k|\Delta_k|$, and $\Delta_{k+1,0} = \Delta_k$. One can see that $|\mathcal{T}_{k+1,2j}|$ is bounded by either $2k|\Delta_k| = 2k|\Delta_{k+1,0}|$ (if $j=0$) or $2|\Delta_{k+1,j}|$ $(\le 2k|\Delta_{k+1,j}|)$. Furthermore, $|\mathcal{T}_{k+1,2j+1}|$ is bounded by $|\Delta_{k+1,j}|$, the number of the latest verified delta set. Hence we have

$$|\mathcal{T}_{k+1,2j}| + |\mathcal{T}_{k+1,2j+1}| \le (2k+1)|\Delta_{k+1,j}|,$$

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

Now we only need to show that the number of all the previously ignored terms, denoted by $\mathcal{T}'_{k+1}$, is bounded by $|\Delta_{k+1}|$. Suppose $\mathcal{T}'_{k+1} = \bigcup_{i\in S} \mathcal{T}'_{k+1,i}$, where S is a set of indexes with $|S| \le m$ in (16), and

$$\mathcal{T}'_{k+1,i} := \{\boldsymbol{u} \in \mathcal{T}'_{k+1} : \boldsymbol{u} = (u_1,\ldots,u_k,i)\}.$$

Then for each i, $|\mathcal{T}'_{k+1,i}|$ is bounded by the number of the latest verified delta set, say $|\Delta_{k+1,p_i}|$. Thus the conclusion can be proved if one notices $\bigcup_{i\in S} \Delta_{k+1,p_i} \subseteq \Delta_{k+1}$. □

Theorem 4.5. *Assume that $T_1,\ldots,T_n$ are constructed. The complexity for Algorithm 5 to complete the change of ordering is bounded by $O(nD(N+\hat{N}\overline{N}D))$, where N is the maximal number of nonzero entries in the multiplication matrices $T_1,\ldots,T_n$, and $\hat{N}$ and $\overline{N}$ are respectively the number of polynomials and the maximal term number of all polynomials in the resulting Gröbner basis.*

4.4. Example

Consider the ideal $I \subset \mathbb{F}_{65521}[x_1,x_2]$ defined by its DRL Gröbner basis $(x_1 < x_2)$

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

Now we compute the Gröbner basis G_2 of I w.r.t. LEX with Algorithm 5.

The ideal $\mathbb{F}_{65521}[x_1,x_2]/\langle G_1\rangle$ is of dimension 12. In Line 1 of Algorithm 5 the canonical basis and further the multiplication matrices T_1 and T_2 are computed accordingly.

In Line 2 a vector is generated at random as

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

After the initialization of $\boldsymbol{u}$, F, G, and E at Line 3, the main loop begins to handle terms one by one according to the LEX ordering. In Lines 5 and 6, the value e at the handled term $\boldsymbol{u}$ is computed

Table 1
Example for the BMS-based method.

Term $\boldsymbol{u}$	$\Delta(\boldsymbol{u})$	F: polynomial set valid up to $\boldsymbol{u}$
$(0,0)$	$(0,0)$	x_1, x_2
$(1,0)$	–	$x_1 + 65437, x_2$
$(2,0)$	$(0,0), (1,0)$	$x_1^2 + 65437\,x_1 + 21672, x_2$
$(3,0)$	–	$x_1^2 + 62681\,x_1 + 41493, x_2$
$(4,0)$	$(0,0), (1,0), (2,0)$	$x_1^3 + 62681\,x_1^2 + 35812\,x_1 + 18557, x_2$
$(5,0)$	–	$x_1^3 + 30688\,x_1^2 + 45566\,x_1 + 54643, x_2$
$(6,0)$	$(0,0), (1,0), (2,0), (3,0)$	$x_1^4 + 30688\,x_1^3 + 20026\,x_1^2 + 45766\,x_1 + 5434, x_2$
$(7,0)$	–	g_1, x_2
$(0,1)$	–	$g_1, x_2 + 65034\,x_1^3 + 24330\,x_1^2 + 14876\,x_1 + 52361$
$(1,1)$	–	$g_1, x_2 + 64550\,x_1^3 + 37707\,x_1^2 + 48745\,x_1 + 7628$
$(2,1)$	–	$g_1, x_2 + 38842\,x_1^3 + 5603\,x_1^2 + 45755\,x_1 + 44311$
$(3,1)$	–	$g_1, x_2 + 9449\,x_1^3 + 20826\,x_1^2 + 39078\,x_1 + 38885$
$(0,2)$	$(0,0), (1,0), (2,0), (3,0), (0,1)$	$g_1, x_2^2 + 38885\,x_2 + 65360\,x_1^3 + 1782\,x_1^2 + 36000\,x_1 + 39469$ $x_2x_1 + 20826\,x_1^3 + 28385\,x_1^2 + 55917\,x_1 + 37174$
$\vdots$	$\vdots$	$\vdots$

with the randomly chosen vector $\boldsymbol{r}$ and added to the maintained list E of computed values. Then BMSUpdate() is applied to the current truncated mapping to compute the new polynomial set F valid up to $\boldsymbol{u}$ in Line 7, followed by reduction applied to F in Line 10.

For the first terms $\boldsymbol{u}$s in the iterations, the polynomial sets F valid up to $\boldsymbol{u}$s, together with the corresponding delta sets $\Delta(\boldsymbol{u})$, are recorded in Table 1. For example, at the term $(4,0)$, the polynomial $x_1^2 + 62681\,x_1 + 41493 \in F$, which is valid up to $(3,0)$, is no longer 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 $x_1^3 + 62681\,x_1^2 + 35812\,x_1 + 18557$ in it is valid up to $(4,0)$.

As the iterations proceed, the first polynomial in G_2:

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

is obtained at the term $(7,0)$. 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:

$$g_2 = x_2^3 + 7\,x_1^2x_2^2 + 15\,x_1^2x_2 + 2\,x_1^3 + 9$$

is obtained at $(3,5)$. Now the main loop of Algorithm 5 ends.

In Line 13 we find that both polynomials g_1 and g_2 satisfy the condition (1) and

$$\dim(\mathbb{F}_{65521}[x_2, x_1]/\langle g_1, g_2\rangle) = 12.$$

Hence IsGB(F) returns true and the output is $G_2 = \{g_1, g_2\}$.

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. According to our experiments, 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 though the base field itself is quite large and different random vectors $\boldsymbol{r}$ are tried. The failure of the BMS algorithm may be due to the special structure of the ideal $\langle G\rangle$ (e.g., roots of this polynomial systems are of multiplicities) so that the equality $I(E) = \langle G\rangle$ does not hold for most random vectors. The further investigation on when the BMS fails is our future work. In particular, we would like to point out that an FGLM-like algorithm has been proposed in Berthomieu et al. (2015) (Algorithm 3) to compute the Gröbner basis of $I(E)$ for a mapping E. The computation in this algorithm is essentially reduced to that of the kernels of multi-Hankel matrices, which seem to be related to Gorenstein algebras (Lasserre et al., 2013).

5. Multiplication matrix T_1: sparsity and complexity

In the previous description and complexity analyses of all the algorithms, the multiplication matrices $T_1, \ldots, T_n$ are assumed known. In this section, for generic polynomial systems and the term ordering DRL, the multiplication matrix T_1 is exploited, on its sparsity and cost for construction. We are able to give an explicit formula to compute the number of dense columns in T_1, and we also analyze the asymptotic behavior of this number, which further leads to a finer complexity analysis for the change of ordering for generic systems. The term ordering is preassigned as DRL in this section without further notification.

5.1. Construction of multiplication matrices

Given the Gröbner basis G of a 0-dimensional ideal I w.r.t. DRL, let $B = [\epsilon_1, \ldots, \epsilon_D]$ be the canonical basis of $\mathbb{K}[x_1, \ldots, x_n]/\langle G\rangle$, and $L := \{\mathrm{lt}(g) : g \in G\}$. The three cases of the multiplication $\epsilon_i x_j$ for the construction of the ith column of T_j in FGLM are reviewed below (Faugère et al., 1993).

(a) The term $\epsilon_i x_j$ is in B: the coordinate vector of NormalForm$(\epsilon_i x_j)$ is $(0, \ldots, 0, 1, 0, \ldots, 0)^t$, where the position of 1 is the same as that of $\epsilon_i x_j$ in B;
(b) The term $\epsilon_i x_j$ is in L: the coordinate vector can be obtained easily from the polynomial $g \in G$ such that $\mathrm{lt}(g) = \epsilon_i x_j$;
(c) Otherwise: the normal form of $\epsilon_i x_j$ w.r.t. G has to be computed to get the coordinate vector.

Obviously, the ith column of T_j is sparse if case (a) occurs, thus a dense column can only come from cases (b) and (c). Furthermore, the construction for a column will not be free of arithmetic operations only if that column belongs to case (c). As a result, we are able to connect the cost for constructing the multiplication matrices with the numbers of dense columns in them.

Proposition 5.1. *Denote by M_i the number of dense columns in the multiplication matrix T_i. Then the matrices $T_1, \ldots, T_n$ can be computed within $O(D^2 \sum_{i=1}^n M_i)$ arithmetic operations.*

Proof. Direct result from the proof of Proposition 3.1 in Faugère et al. (1993). □

As shown in Section 3, among all the multiplication matrices, T_1 is the most important one, and it is also of our main interest. However, for an arbitrary ideal, currently we are not able to analyze the cost to construct T_1 by isolating it from the others in Proposition 5.1, for the analysis on T_1 needs information from the other matrices too.

In the following parts we first focus on generic sequences which impose stronger conditions on T_1 so that the analyses on it become feasible. We show that the construction of T_1 for generic sequences is free and present finer complexity results based on an asymptotic analysis.

5.2. Structures of term ideals of generic sequences

Let $P = [f_1, \ldots, f_n]$ be a sequence of polynomials in $\mathbb{K}[x_1, \ldots, x_n]$ of degree $d_1, \ldots, d_n$. If $d_1 = \cdots = d_n = d$, we call it a sequence of degree d. We are interested in the properties of the multiplication matrices for the ideal generated by P if $f_1, \ldots, f_n$ are chosen "at random". Such properties can be regarded generic in all sequences. More precisely, let U be the set of all sequences of n polynomials of degree $d_1, \ldots, d_n$, viewed as an affine space with the coefficients of the polynomials in the sequences as the coordinates. Then a property of such sequences is *generic* if it holds on a Zariski-open in U. Next for simplicity, we will say some property holds "for a generic sequence" if it is a generic one, and also P is "a generic sequence" if its properties of our interest are generic.

For a generic sequence $[f_1, \ldots, f_n]$, the term ideal $\mathrm{lt}(\langle f_1, \ldots, f_n\rangle)$ is the same as $\mathrm{lt}(\langle f_1^h, \ldots, f_n^h\rangle)$, where f_i^h is homogenized polynomial of f_i for the highest degree (Bardet et al., 2015). That is to say, we only need to study homogeneous generic sequences. Hence in the following part of this section, a generic sequence is further assumed homogeneous.

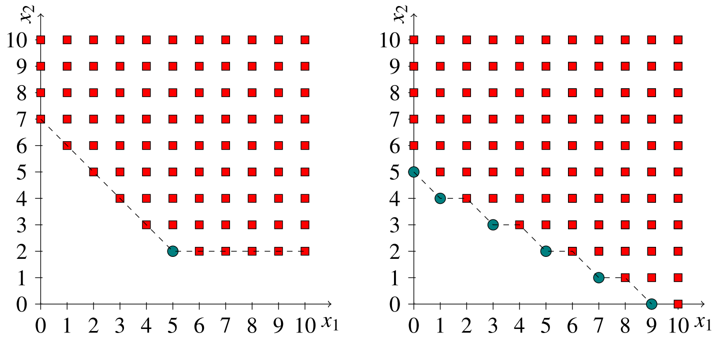

Fig. 2. One term $\boldsymbol{u} \in \mathrm{lt}(\langle P \rangle)$ (●) and the terms it determines (🟥) (left). Minimal generators of $\mathrm{lt}(\langle P \rangle)$ (●) and terms in $\mathrm{lt}(\langle P \rangle)$ (🟥) (right).

Since we restrict to the situation where the number of polynomials is equal to that of variables, a generic sequence is a *regular* one (Lazard, 1983). We first recall the well-known characterization of a regular sequence via its Hilbert series.

Theorem 5.2. *Let $[f_1, \ldots, f_r]$ be a sequence in $\mathbb{K}[x_1, \ldots, x_n]$ with $\deg(f_i) = d_i$. Then it is regular if and only if its Hilbert series is*

$$\frac{\prod_{i=1}^{r}(1 - z^{d_i})}{(1 - z)^n}.$$

Let P be a generic sequence of degree d. Then we know its Hilbert series is

$$H(n, d) := (1 - z^d)^n/(1 - z)^n = (1 + z + z^2 + \cdots + z^{d-1})^n, \qquad (17)$$

from which one can easily derive that the degree of $\langle P \rangle$ is d^n, and that the greatest total degree of terms in the canonical basis is $(d - 1)n$.

Proposition 5.3. *Let $\mathbb{K}$ be an infinite field and $P = [f_1, \ldots, f_n]$ be a generic sequence in $\mathbb{K}[x_1, \ldots, x_n]$ with $\deg(f_i) = d_i$. Then for any $\boldsymbol{t} \in \mathrm{lt}(\langle P \rangle)$, if x_i divides $\boldsymbol{t}$, then $\frac{\boldsymbol{t}x_j}{x_i} \in \mathrm{lt}(\langle P \rangle)$ for all $j > i$.*

Proof. Straightforward from Eisenbud (1995, Thm. 15.20) and Miller and Sturmfels (2005, Prop. 2.3). □

Proposition 5.3 imposes a stronger requirement on the structure of the terms in $\mathrm{lt}(\langle P \rangle)$ for a generic sequence P. For the bivariate case, once a term $\boldsymbol{u}$ is known to be an element in $\mathrm{lt}(\langle P \rangle)$, the terms in $\mathrm{lt}(\langle P \rangle)$ determined by it are illustrated in Fig. 2 (left), and furthermore, in the right figure the shape all terms in $\mathrm{lt}(\langle P \rangle)$ form.

The structures of term ideals of generic sequences, also called *generic initial ideals*, have been studied in Galligo (1974), Bayer and Stillman (1987), Pardue (1994), and interested readers may also refer to Eisenbud (1995, Sec. 15.9) and Miller and Sturmfels (2005, Sec. 2.2).

Corollary 5.4. *Let $\mathbb{K}$ be an infinite field, $P \in \mathbb{K}[x_1, \ldots, x_n]$ be a generic sequence of degree d, and B be the canonical basis of $\mathbb{K}[x_1, \ldots, x_n]/\langle P \rangle$ w.r.t. DRL. Denote by $B(k)$ the set of terms of total degree k in B. Then for $k = 1, \ldots, (d - 1)n$, $B(k)$ consists of the first $|B(k)|$ smallest terms in all terms of total degree k.*

Proof. In Proposition 5.3, $\frac{\boldsymbol{t}x_j}{x_i}$ is smaller than $\boldsymbol{t}$ w.r.t. DRL for $j > i$, and the corollary follows. □

5.3. Sparsity and construction of T_1

Let $P \subset \mathbb{K}[x_1, \dots, x_n]$ be a generic sequence and G be the Gröbner basis of $\langle P \rangle$. Then polynomials in G can be assumed dense (in the case when $\mathbb{K}$ is of characteristic 0 or of large cardinality). As the number of dense columns in T_1 will directly lead to a bound on the number of nonzero entries in T_1, the study of T_1 sparsity is reduced to that of how many cases of (b) and (c) happen. Combining the Hilbert series of a generic sequence and Corollary 5.4, we are able to count the dense columns in T_1.

Proposition 5.5. *Let $\mathbb{K}$, P, B, and $B(k)$ be the same as those in Corollary 5.4. Then the number of dense columns in the multiplication matrix T_1 is equal to the greatest coefficient in the expansion of $(1 + z + \cdots + z^{(d-1)})^n$.*

Proof. Let $k' = (d-1)n$, and denote by $\mathscr{T}^{(k)}$ the set of all terms in $\mathbb{K}[x_1, \dots, x_n]$ of total degree k.

Suppose that $\boldsymbol{u}$ is the lth smallest term in $\mathscr{T}^{(k)}$. Then $x_1\boldsymbol{u}$ is also the lth smallest term in $\mathscr{T}^{(k+1)}$. Hence by Corollary 5.4, if $|B(k)| \le |B(k+1)|$, then for every $\boldsymbol{u} \in B(k)$, $x_1\boldsymbol{u}$ is still in $B(k+1)$. Therefore it belongs to case (a) we reviewed in Section 5.3, and the corresponding column in T_1 is a sparse one. If $|B(k)| > |B(k+1)|$, we will have $|B(k)| - |B(k+1)|$ dense columns which come from the fact that they belong to case (b) or (c).

As the coefficients in the expansion of $(1+z+\cdots+z^{(d-1)})^n$ are symmetric to the central coefficient (or the central two when $(d-1)n$ is odd), the condition $|B(k)| > |B(k+1)|$ holds for the first time when $k = k_0$, the index of the central term (or of the second one in the central two terms). Then the number of dense columns is

$$\begin{aligned}&(|B(k_0)| - |B(k_0+1)|) + (|B(k_0+1)| - |B(k_0+2)|)\\ &+ \cdots + (|B(k'-1)| - |B(k')|) + |B(k')| = |B(k_0)|.\end{aligned}$$

That ends the proof, for such a coefficient $|B(k_0)|$ is exactly the greatest one. $\square$

The Hilbert series is usually used to analyze the behaviors of Gröbner basis computation, for example the regularity of the input ideal. As the leading terms of polynomials in the Gröbner basis and the canonical basis determine each other completely, it is also natural to have Proposition 5.5, which links the canonical basis and Hilbert series.

Remark 5.1. When $d = 2$, the number of dense columns in T_1 is the binomial coefficient $C_n^{k_0}$, where

$$k_0 = \begin{cases} n/2 & \text{if } n \text{ is even;} \\ \frac{n+1}{2} & \text{if } n \text{ is odd.} \end{cases}$$

For the case $d = 3$, the greatest coefficient is the *central trinomial coefficient*.

Corollary 5.6. *The percentage of nonzero entries in the multiplication matrix T_1 for a generic sequence of degree d is bounded by $(m_0 + 1)/D$, where m_0 is the number of dense columns computed from Proposition 5.5.*

Proof. The number of nonzero entries in the dense columns is bounded by $m_0 D$, and that in the other columns is smaller than D. $\square$

With Proposition 5.3, we can take a step forward from Proposition 5.5. That is, we show that case (c) will not occur during the construction of T_1.

Proposition 5.7. *Follow the notations in Proposition 5.3. Then for any term $\boldsymbol{u} \notin \mathrm{lt}(\langle P \rangle)$, $x_1\boldsymbol{u}$ is either not in $\mathrm{lt}(\langle P \rangle)$ or a minimal generator of $\mathrm{lt}(\langle P \rangle)$.*

Proof. Suppose that $x_1\boldsymbol{u} = (u_1, \dots, u_n) \in \mathrm{lt}(\langle P \rangle)$ is not a minimal generator. We will draw a contradiction by showing $\boldsymbol{u} \in \mathrm{lt}(\langle P \rangle)$.

Without loss of generality, we can assume that each $u_i \neq 0$ for $i = 1, \dots, n$, otherwise we can reduce to the $n-1$ case by ignoring the ith component of $\boldsymbol{u}$ and repeat this process if necessary. As $x_1\boldsymbol{u}$ is not a minimal generator, there exists an integer k $(1 \le k \le n)$ such that $\boldsymbol{u}^{(k)} := (u_1, \dots, u_k - 1, \dots, u_n) \in \mathrm{lt}(\langle P\rangle)$. Then when $k = 1$, $\boldsymbol{u} = \boldsymbol{u}^{(k)} \in \mathrm{lt}(\langle P\rangle)$; otherwise, $\boldsymbol{u} = \frac{\boldsymbol{u}^{(k)} x_k}{x_1} \in \mathrm{lt}(\langle P\rangle)$ by Proposition 5.3. $\square$

Corollary 5.8. *The number of dense columns in T_1 for a generic sequence $P \subset \mathbb{K}[x_1, \dots, x_n]$ is equal to the cardinality of $\{g \in G_1 : x_1 \,|\, \mathrm{lt}(g)\}$, where G_1 is the Gröbner basis of $\langle P\rangle$ w.r.t. DRL.*

Remark 5.2. By Corollary 5.8, for a generic sequence, to construct T_1 one only needs to find the leading term of which polynomial in G_1 is a given term $x_1\boldsymbol{u}$. Thus we can conclude that the construction of T_1 is free of arithmetic operations. Even for real implementations, the cost for constructing T_1 is also quite small compared with that for the change of ordering. Bearing in mind that the ideal generated by a generic sequence is in shape position, we know the complexity in Theorem 3.2 is indeed the complete complexity for the change of ordering for generic sequences, including construction of T_1, the only multiplication matrix needed.

5.4. Asymptotic analysis

Next we study the asymptotic behavior of the number of dense columns in T_1 for a generic sequence of degree d, with n fixed and d increasing to $+\infty$. These results are mainly derived from a more detailed asymptotic analysis of coefficients of the Hilbert series of semi-regular systems in Bardet (2004), where standard methods in asymptotic analysis, like the saddle-point and coalescent saddle points methods, are applied. However, in Bardet (2004) the asymptotic estimation of $I_d(m)$ as defined below is studied for the case when d is fixed (as 2, and thus it is the quadratic case) and n tends to $+\infty$, while here we are interested in the dual case when n is fixed and d tends to $+\infty$.

The target of this subsection is to find the dominant term of the greatest coefficient in the expansion of the Hilbert series $H(n,d)$ in (17), as d tends to $+\infty$ and n is fixed. First one writes the mth coefficient $I_d(m)$ of $H(n,d)$ with the Cauchy integration:

$$I_d(m) = \frac{1}{2\pi\,\mathrm{i}} \oint \frac{H(n,d)(z)}{z^{m+1}} dz = \frac{1}{2\pi\,\mathrm{i}} \oint \frac{(1-z^d)^n}{(1-z)^n z^{m+1}} dz.$$

With $F(z) := \frac{(1-z^d)^n}{(1-z)^n z^{m+1}} = e^{f(z)}$ and $g(z) := 1$, $I_d(m)$ becomes the form convenient to the asymptotic analysis

$$I_d(m) = \frac{1}{2\pi\,\mathrm{i}} \oint g(z) e^{f(z)} dz.$$

Suppose the greatest coefficient in $H(n,d)$ comes from the m_0th term. Since we are interested in the asymptotic behavior, we can assume $m_0 = (d-1)n/2$. As a special case of Bardet (2004, Lem. 4.3.1), we have the following result.

Proposition 5.9. *Suppose $m_0 = (d-1)n/2$. Then the dominant term of $I_d(m_0)$ is*

$$I_d(m_0) \sim \sqrt{\frac{1}{2\pi f''(r_0)}}\, e^{f(r_0)},$$

where $f(z) = n\log(\frac{1-z^d}{1-z}) - (m+1)\log(z)$, and r_0 is the positive real root of $f'(z)$. Furthermore, r_0 tends to 1 as d increases to $+\infty$.

To prove the fact that the positive real root r_0 of $f'(z)$ tends to 1, one needs to use the equality $m_0 = (d-1)n/2$. Other parts of the proof are the same as those in Bardet (2004, Sec. 4.3.2).

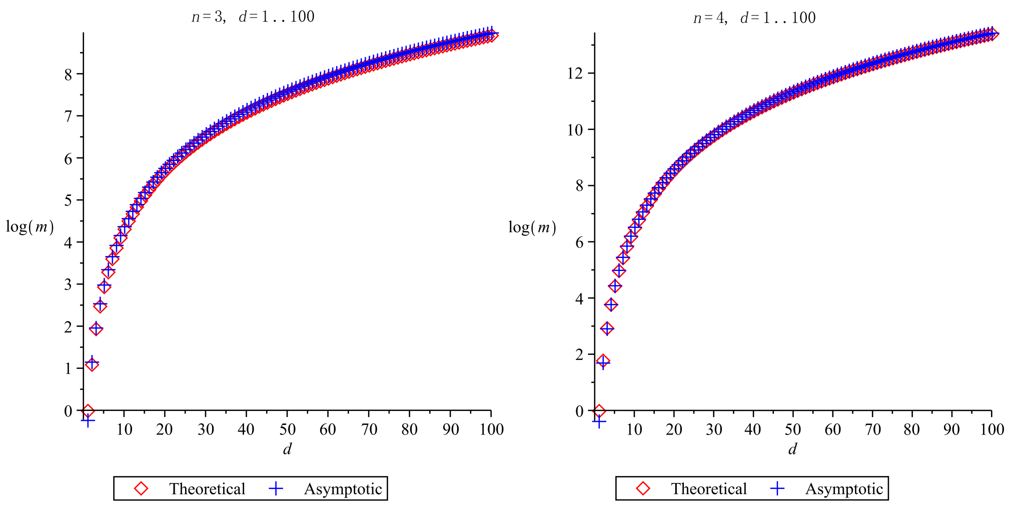

Fig. 3. Number of dense columns in T_1 for $n = 3, 4$ and $d = 1, \dots, 100$.

Next we investigate the value of $f''(r_0)$ and $F(r_0)$ in the dominant part of $I_d(m_0)$ as r_0 tends to 1. Set $h(z) := \frac{1-z^d}{1-z} = 1 + z + \cdots + z^{d-1}$. Then

$$f''(z) = n\frac{h''(z)h(z) - h'(z)^2}{h(z)^2} + \frac{d+1}{z}.$$

Noting that $h(1) = d$, $h'(1) = d(d-1)/2$, and $h''(1) = d(d-1)(d-2)/3$, we have

$$f''(1) = nd^2/12 + O(d).$$

With the easily obtained the equality $F(1) = d^n$, we have the following asymptotic estimation of $I_d(m_0)$.

Corollary 5.10. *Let n be fixed. As d tends to* $+\infty$, $I_d(m_0) \sim \sqrt{\frac{6}{n\pi}} d^{n-1}$.

This asymptotic estimation of the greatest coefficient in $H(n, d)$ accords with the theoretical one. Fig. 3 shows the number of dense columns derived from both Proposition 5.5 and Corollary 5.10. As can be shown from this figure, the asymptotic estimation is good, even when d is small.

Corollary 5.11. *Let n be fixed. As d tends to* $+\infty$, *the following statements hold:*

(a) *the percentage of nonzero entries in* T_1 *is* $\sim \sqrt{\frac{6}{n\pi}}/d$;
(b) *for a generic sequence of degree d, the complexity in Theorem 3.2 is* $O(\sqrt{\frac{6}{n\pi}} D^{2+\frac{n-1}{n}})$.

As Corollary 5.11 shows, for a generic sequence, the multiplication matrix T_1 becomes sparser as d increases. Furthermore, the complexity of Algorithm 3 is smaller in both the exponent and constant compared with FGLM.

6. Experiments

The first method for the shape position case, namely Algorithm 3, has been implemented in C over fields of characteristic 0 and finite fields. The BMS-based method for the general case has been

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

Name	D	Matrix density			FGb		Magma		Singular		Speedup	
		Actual	Theoretical	Asymp.	F_5(C)	Sparse FGLM	F_4	FGLM	Buchberger	FGLM	Magma	Singular
Random 11, $d=2$	2^{11}	21.53%	22.56%	20.83%	3.3 s	1.7 s	18.0 s	162.2 s	623.9	328.6	95.4	193.3
Random 12, $d=2$	2^{12}	21.26%	22.56%	19.95%	19.9 s	10.7 s	134.9	1335.8	4867.4	2581.1	124.8	241.2
Random 13, $d=2$	2^{13}	19.98%	20.95%	19.16%	118.0 s	80.8 s	949.6	10757.4	36727.0	19820.2	133.1	245.3
Random 14, $d=2$	2^{14}	19.64%	20.95%	18.47%	747.2 s	559.0 s	7832.4	84374.6			150.9	
Random 15, $d=2$	2^{15}	18.52%	19.64%	17.84%	5364.6 s	3894.6 s						
Random 3, $d=15$	3375	4.46%	5.01%	5.32%	0.7 s	1.8 s	0.95 s	262.9 s	8.3 s	1067.7 s	146.1	593.2
Random 3, $d=16$	4096	4.17%	4.69%	4.99%	1.1 s	2.9 s	1.5 s	333.8 s	13.1 s	1904.6 s	115.1	656.8
Random 3, $d=17$	4913	3.92%	4.42%	4.69%	1.5 s	4.4 s	1.95 s	585.3 s			133.0	
Random 3, $d=18$	5832	3.70%	4.17%	4.43%	2.3 s	6.5 s	2.9 s	1142.6 s			175.8	
Random 3, $d=20$	8000	3.32%	3.75%	3.99%	4.3 s	13.9 s						
Random 3, $d=22$	10648	3.01%	3.41%	3.63%	8.1 s	27.9 s						
Random 3, $d=24$	13824	2.76%	3.13%	3.32%	14.1 s	57.0 s						
Random 3, $d=30$	27000	2.20%	2.50%	2.66%	65.9 s	325.7 s						
Random 3, $d=40$	64000	1.64%	1.88%	1.99%	470.1 s	3491.5 s						
Random 4, $d=8$	4096	7.54%	8.40%	8.64%	2.5 s	4.2 s	3.5 s	556.7 s	70.1 s	1994.0 s	132.5	474.8
Random 4, $d=9$	6561	6.66%	7.45%	7.68%	6.6 s	13.5 s	9.3 s	1800.9 s			133.4	
Random 4, $d=10$	10000	5.97%	6.70%	6.91%	15.5 s	45.7 s						
Random 4, $d=11$	14641	5.40%	6.09%	6.28%	36.8 s	127.0 s						
Random 4, $d=12$	20736	4.94%	5.57%	5.76%	81.3 s	363.6 s						
Random 4, $d=13$	28561	4.55%	5.14%	5.32%	182.0 s	713.9 s						
Katsura 11	2^{11}	21.53%	22.56%	20.83%	4.7 s	1.7 s	18.3 s	178.6 s	632.0 s	328.4 s	105.1	193.2
Katsura 12	2^{12}	21.26%	22.56%	19.95%	29.6 s	10.6 s	147.9 s	1408.1 s	5061.8 s	2623.5 s	132.8	247.5
Katsura 13	2^{13}	19.86%	20.95%	19.16%	177.2 s	80.9 s	1037.2 s	10895.4 s			134.7	
Katsura 14	2^{14}	19.64%	20.95%	18.47%	1285.1 s	553.4 s	9599.0 s	87131.9 s			157.4	
Katsura 15	2^{15}	18.52%	19.64%	17.84%	8487.0 s	3874.6 s						
Eco 12 ‡	1024	29.69%	NA	NA	2.8 s	0.35 s	29.6 s	456.5 s	614.7 s	38.9 s	1304	111.1
Eco 13 ‡	2048	27.52%	NA	NA	15.3 s	2.0 s	262.7 s	5692.7 s			2846	
Eco 14 ‡	4096	26.4%	NA	NA	85.2 s	13.0 s						
Eco 15 ‡	8192	24.9%	NA	NA	587.0 s	96.3 s						
Crit $D=2, p=4, n=9$	896	30.17%	NA	NA	0.5 s	0.24 s	3.0 s	17.4 s	95.2 s	32.9 s	72.5	137.1
Crit $D=2, p=4, n=10$	1344	31.13%	NA	NA	1.6 s	0.68 s	11.8 s	62.1 s			91.3	
Crit $D=2, p=4, n=11$	1920	31.86%	NA	NA	4.3 s	1.77 s	40.7 s	192.5 s			108.8	
Crit $D=3, p=3, n=6$	2160	17.52%	NA	NA	1.3 s	1.4 s	4.3 s	134.2 s	140.5 s	340.6 s	95.9	243.3
Crit $D=3, p=3, n=7$	6480	17.39%	NA	NA	26.2 s	34.2 s	122.8 s	4139.4 s			121	
Crit $D=3, p=3, n=8$	18144	17.63%	NA	NA	520.1 s	762.6 s						
Crit $D=4, p=2, n=5$	1728	14.46%	NA	NA	0.48 s	0.70 s	1.4 s	57.3 s	33.5 s	165.4 s	81.9	236.3
Crit $D=4, p=2, n=6$	6480	14.11%	NA	NA	16.3 s	27.3 s	63.2 s	3196.7 s			117.1	
Crit $D=5, p=2, n=5$	6400	11.00%	NA	NA	10.2 s	19.9 s	27.9 s	2335.1 s			117.3	
Crit $D=6, p=2, n=5$	18000	8.80%	NA	NA	123.4 s	346.7 s						
MinR(9, 6, 3)	980	26.82%	NA	NA	0.7 s	0.3 s	6.3 s	22.7 s	137.5 s	38.1 s	75.7	127.0
MinR(9, 7, 4)	4116	22.95%	NA	NA	17.5 s	10.8 s	208.1 s	1360.4 s	4985.8 s	2490.3 s	126.0	230.6
MinR(9, 8, 5)	14112	19.04%	NA	NA	297.4 s	337.7 s						
MinR(9, 9, 6)	41580	16.91%	NA	NA	5010.7 s	7086.6 s						
DLP EC $n=4$	4096	7.54%	NA	NA	2.4 s	4.25 s	7.4 s	475.8 s	72.4 s	1823.6 s	112.0	429.1
DLP Edwards $n=4$	512	7.68%	NA	NA	0.03 s	0.02 s	0.16 s	21.4 s	8.3 s	5.1 s	1066	255
DLP Edwards $n=5$	2^{16}	3.30%	NA	NA	1861.6 s	5735.9 s						
Cyclic 10 †	34940	1.00%	NA	NA		1107.8 s	>16 hrs and >16 Gig					

implemented preliminarily in Magma over large finite fields. 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 records the timings (in seconds) of our implementations of F_5 and Algorithm 3 applied to benchmarks including theoretical ones like Katsura systems (Katsura n) and randomly generated polynomial systems (Random n, d), and practical ones like MinRank problems from Cryptography (Faugère et al., 2010) and systems coming from economic modelings (Eco n) (Morgan, 1987, pp. 148), critical points computation (Crit D, p, n) (Faugère et al., 2012), and algebraic cryptanalysis of some curve-based cryptosystem (DLP EC and DLP Edwards) (Faugère et al., 2013). In this table, the instances marked with † are indeed not in shape position, and the timings for such instances only indicate those of computing the univariate polynomial in the LEX Gröbner basis.

Table 3
Performances of Algorithm 5 for the general case from DRL to LEX.

Name	n	D	Mat. density	Poly. density	# Passes	# Mat.	# Red.	# Multi.
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	$nD^{2.968}$

Furthermore, for Katsura and randomly generated systems, three kinds of T_1 density are also recorded, namely the actual one (column "Actual"), the theoretical one by Proposition 5.5 (column "Theoretical"), and the asymptotic one by Corollary 5.10 (column "Asymp.").

As shown by this table, the current implementation of Algorithm 3 outperforms the FGLM implementations in Magma and Singular. Take the Random 13, $d = 2$ instance for example, the FGLM implementations in Magma and Singular take 10757.4 and 19820.2 seconds respectively, while the new implementation only needs 80.8 seconds. This is around 132 and 244 times faster. Such an efficient implementation is now able to manipulate ideals in shape position of degree greater than 60000. It is also important to note that with this new algorithm, the time devoted to the change of ordering is somehow of the same order of magnitude as the DRL Gröbner basis computation.

In addition, for Katsura and randomly generated systems, compared to the experimental density of T_1, the theoretical bound works and the asymptotic estimation is also close. For all the benchmarks, T_1 holds a sparse structure to some extent, even when the input ideals are defined by dense polynomial systems like random ones. In fact, dense columns of T_1 only occur at or close to the end of T_1, and most of the other columns are sparse, with only one nonzero component equal to 1, as analyzed in Section 5.3. For matrices with such a structure, 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.

Table 3 illustrates the performances of Algorithm 5 for the general case. As currently this method is only implemented preliminarily in Magma, only the number of field multiplications and other critical 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. The first 4 columns "n", "D", "Mat. density", and "Poly. density" present the number of variables, the degree of the input ideal, the maximal percentage of nonzero entries in the matrices $T_1, \dots, T_n$, and the density of resulting Gröbner bases, while in the next 4 columns the numbers of passes in the main loop of Algorithm 5, matrix multiplications, reductions and field multiplications are recorded.

As one can see from this table, the numbers of passes accord with the bound derived in Theorem 4.1, and the number of operations is less than the original FGLM algorithm for Cyclic-like benchmarks. However, for instances of 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. Fortunately, in the top-level algorithm (Algorithm 1), it is not common to handle such ideals in shape position with this method.

Acknowledgments

The authors would like to thank Daniel Augot for his very helpful comments on the incremental Wiedemann algorithm and literatures about the BMS algorithm and the reviewers for their very insightful and detailed comments which lead to considerable improvements on the paper. This work was partially supported by the HPAC grant of the French National Research Agency (HPAC ANR-11-BS02-013), by National Natural Science Foundation of China (NSFC 11401018), and by the ECCA project of the Sino-French Laboratory for Computer Science, Automation and Applied Mathematics.

Appendix A. BMS algorithm for LEX

For an integer r chosen by the user for the termination criterion, we are interested in what happens when the polynomial set F in the BMS algorithm keeps unchanged for r iterations after handling some term $\boldsymbol{u}$. Let $\boldsymbol{u} = (u_1, \dots, u_n) \in \mathbb{Z}^{(1\times n)}$ be a term. Then for each i $(0 \le i \le n-1)$, denote

$$\boldsymbol{u}_i = (u_{i+1}, \dots, u_n) \in \mathbb{Z}^{(1\times(n-i))}.$$

For a term $\boldsymbol{u}$, if for each $i = 0, \dots, k$ $(0 \le k \le n-1)$, the polynomial set F does not change for the first r terms in $\{\boldsymbol{v} \in \mathbb{Z}^{(1\times n)} : \boldsymbol{v}_i \ne \boldsymbol{u}_i\}$, then we say that the BMS algorithm has a kth jump after $\boldsymbol{u}$. The occurrence of a kth jump means that in general the algorithm has finished the computation for $\mathbb{K}[x_1, \dots, x_k]$ with a large probability if the ground field $\mathbb{K}$ is of large cardinality.

In the algorithm we present, the variable "count" is for recording the number of iterations with which the polynomial set F keeps unchanged, "layer" is for recording the index k of x_k with a kth jump already occurred, and "layerLift" is a Boolean value for recording whether to add the index to $k+1$ when some condition is satisfied.

Algorithm 6: BMS algorithm for LEX $F := \mathsf{BMSlex}(E)$.

```
Input: $n$-dimensional mapping $E$
Output: Gröbner basis of $I(E)$ w.r.t. LEX with a large probability of success
count := 0;  layer := 0;  layerLift := true;
$\boldsymbol{u} := \boldsymbol{1}$;  $F := \{1\}$;
repeat
    $F_2 := \mathsf{BMSUpdate}(F, \boldsymbol{u}, E)$;
    if $F_2 = F$ then
        count := count + 1;
    else
        count := 0;  layerLift := false;
    end
    if count = $r$ then
        count := 0;
        if layerLift = true then
            layer := layer + 1;
            if layer = $n$ then
                return $F$
            end
        else
            layer := 1;  layerLift := true;
        end
    end
    $\boldsymbol{u}$ := the smallest term $\boldsymbol{v}$ which is greater than $\boldsymbol{u}$ w.r.t. LEX and $\boldsymbol{v}_{\text{layer}} \ne \boldsymbol{u}_{\text{layer}}$;
    $F := F_2$;
until 1 = 0;
```

References

Bardet, M., 2004. Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie. Ph.D. thesis. Université Paris VI.

Bardet, M., Faugère, J.-C., Salvy, B., 2015. On the complexity of the F5 Gröbner basis algorithm. J. Symb. Comput. 70, 49–70.

Basiri, A., Faugère, J.-C., 2003. Changing the ordering of Gröbner bases with LLL: case of two variables. In: Sendra, J. (Ed.), Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation. ACM Press, pp. 23–29.

Bayer, D., Stillman, M., 1987. A theorem on refining division orders by the reverse lexicographic order. Duke Math. J. 55 (2), 321–328.

Becker, E., Mora, T., Marinari, M., Traverso, C., 1994. The shape of the shape lemma. In: Giesbrecht, M. (Ed.), Proceedings of the 1994 International Symposium on Symbolic and Algebraic Computation. ACM Press, pp. 129–133.

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

Berthomieu, J., Boyer, B., Faugère, J.-C., 2015. Linear algebra for computing Gröbner bases of linear recursive multidimensional sequences. In: Yokoyama, K., Linton, S., Robertz, D. (Eds.), Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation. ACM Press, pp. 61–68.

Bras-Amorós, M., O'Sullivan, M., 2006. The correction capability of the Berlekamp–Massey–Sakata algorithm with majority voting. Appl. Algebra Eng. Commun. Comput. 17 (5), 315–335.
Brent, R.P., Gustavson, F.G., Yun, D.Y.Y., 1980. Fast solution of Toeplitz systems of equations and computation of Padé approximants. J. Algorithms 1 (3), 259–295.
Buchberger, B., 1985. Gröbner bases: an algorithmic method in polynomial ideal theory. In: Bose, N. (Ed.), Multidimensional Systems Theory. D. Reidel Publishing Company, Dordrecht, pp. 184–232.
Buchmann, J., Pyshkin, A., Weinmann, R.-P., 2006. A zero-dimensional Gröbner basis for AES-128. In: Robshaw, M. (Ed.), Fast Software Encryption. Springer, Berlin/Heidelberg, pp. 78–88.
Collart, S., Kalkbrener, M., Mall, D., 1997. Converting bases with the Gröbner walk. J. Symb. Comput. 24 (3–4), 465–469.
Cox, D., Little, J., O'Shea, D., 1998. Using Algebraic Geometry. Springer-Verlag.
Dahan, X., Jin, X., Moreno Maza, M., Schost, E., 2008. Change of order for regular chains in positive dimension. Theor. Comput. Sci. 392, 37–65.
Eisenbud, D., 1995. Commutative Algebra: with a View Toward Algebraic Geometry. Graduate Texts in Mathematics, vol. 150. Springer-Verlag.
Faugère, J.-C., 1999. A new efficient algorithm for computing Gröbner bases (F_4). J. Pure Appl. Algebra 139 (1–3), 61–88.
Faugère, J.-C., 2002. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In: Mora, T. (Ed.), Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation. ACM Press, pp. 75–83.
Faugère, J.-C., Gaudry, P., Huot, L., Renault, G., 2013. Using symmetries in the index calculus for elliptic curves discrete logarithm. J. Cryptol. 27 (4), 595–635.
Faugère, J.-C., Gaudry, P., Huot, L., Renault, G., 2014. Sub-cubic change of ordering for Gröbner basis: a probabilistic approach. In: Nabeshima, K. (Ed.), Proceedings of the 2014 International Symposium on Symbolic and Algebraic Computation. ACM Press, pp. 170–177. Extended version: Polynomial systems solving by fast linear algebra, preprint at arXiv:1304.6039.
Faugère, J.-C., Gianni, P., Lazard, D., Mora, T., 1993. Efficient computation of zero-dimensional Gröbner bases by change of ordering. J. Symb. Comput. 16 (4), 329–344.
Faugère, J.-C., Mou, C., 2011. Fast algorithm for change of ordering of zero-dimensional Gröbner bases with sparse multiplication matrices. In: Leykin, A. (Ed.), Proceedings of the 2011 International Symposium on Symbolic and Algebraic Computation. ACM Press, pp. 115–122.
Faugère, J.-C., Safey El Din, M., Spaenlehauer, P.-J., 2010. Computing loci of rank defects of linear matrices using Gröbner bases and applications to cryptology. In: Watt, S. (Ed.), Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation. ACM Press, pp. 257–264.
Faugère, J.-C., Safey El Din, M., Spaenlehauer, P.-J., 2012. Critical points and Gröbner bases: the unmixed case. In: van der Hoeven, J., van Hoeij, M. (Eds.), Proceedings of the 2012 International Symposium on Symbolic and Algebraic Computation. ACM Press, pp. 162–169.
Feng, G., Rao, T., 1993. Decoding algebraic-geometric codes up to the designed minimum distance. IEEE Trans. Inf. Theory 39 (1), 37–45.
Galligo, A., 1974. A propos du théoreme de préparation de Weierstrass. Ph.D. thesis. Institut de Mathématique et Sciences Physiques de l'Université de Nice, France.
Høholdt, T., van Lint, J., Pellikaan, R., 1998. Algebraic Geometry Codes. Handbook of Coding Theory. Elsevier, Amsterdam.
Jonckheere, E., Ma, C., 1989. A simple Hankel interpretation of the Berlekamp–Massey algorithm. Linear Algebra Appl. 125, 65–76.
Kaltofen, E., Pan, V., 1991. Processor efficient parallel solution of linear systems over an abstract field. In: Proceedings of the Third Annual ACM Symposium on Parallel Algorithms and Architectures. ACM Press, pp. 180–191.
Lasserre, J.-B., Laurent, M., Mourrain, B., Rostalski, P., Trébuchet, P., 2013. Moment matrices, border bases and real radical computation. J. Symb. Comput. 51, 63–85.
Lazard, D., 1983. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In: Computer Algebra. EUROCAL' 83. Springer, pp. 146–156.
Lazard, D., 1992. Solving zero-dimensional algebraic systems. J. Symb. Comput. 13 (2), 117–131.
Loustaunau, P., York, E., 1997. On the decoding of cyclic codes using Gröbner bases. Appl. Algebra Eng. Commun. Comput. 8 (6), 469–483.
Miller, E., Sturmfels, B., 2005. Combinatorial Commutative Algebra. Graduate Texts in Mathematics, vol. 227. Springer.
Morgan, A., 1987. Solving Polynominal Systems Using Continuation for Engineering and Scientific Problems. Prentice-Hall, Englewood Cliffs, NJ.
Mou, C., 2012. Design of termination criterion of BMS algorithm for lexicographical ordering. J. Comput. Appl. 32 (11), 2977–2980 (in Chinese).
Pardue, K., 1994. Nonstandard Borel-fixed ideals. Ph.D. thesis. Brandeis University, US.
Pascal, C., Schost, E., 2006. Change of order for bivariate triangular sets. In: Dumas, J.-G. (Ed.), Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation. ACM Press, pp. 277–284.
Rouillier, F., 1999. Solving zero-dimensional systems through the rational univariate representation. Appl. Algebra Eng. Commun. Comput. 9 (5), 433–461.
Saints, K., Heegard, C., 2002. Algebraic-geometric codes and multidimensional cyclic codes: a unified theory and algorithms for decoding using Gröbner bases. IEEE Trans. Inf. Theory 41 (6), 1733–1751.
Sakata, S., 1988. Finding a minimal set of linear recurring relations capable of generating a given finite two-dimensional array. J. Symb. Comput. 5 (3), 321–337.
Sakata, S., 1990. Extension of the Berlekamp–Massey algorithm to N dimensions. Inf. Comput. 84 (2), 207–239.
Von zur Gathen, J., Gerhard, J., 2003. Modern Computer Algebra. Cambridge University Press.
Wang, D., 2001. Elimination Methods. Springer-Verlag.
Wiedemann, D., 1986. Solving sparse linear equations over finite fields. IEEE Trans. Inf. Theory 32 (1), 54–62.