Iranian Journal of Mathematical Sciences and Informatics
Vol. 19, No. 1 (2024), pp 117-133
10.61186/ijmsi.19.1.117

A Parametric F_4 Algorithm

Mahdi Dehghani Darmian[a,b*], Amir Hashemi[b,c]

[a]Department of Mathematics, Technical and Vocational University (TVU), Tehran, Iran
[b]School of Mathematics, Institute for Research in Fundamental Sciences (IPM), Tehran, 19395-5746, Iran
[c]Department of Mathematical Sciences, Isfahan University of Technology, Isfahan, 84156-83111, Iran

E-mail: m.dehghanidarmian@ipm.ir; m.dehghanidarmian@gmail.com
E-mail: amir.hashemi@ipm.ir

Abstract. In this paper, we present a parametric F_4 algorithm (so-called PF_4) which can be considered as a generalization of Faugère's F_4 algorithm [8] to polynomial ideals with parametric coefficients. Our approach is based on the F_4 algorithm, Montes DisPGB algorithm [21] and the parametric linear algebra method developed in [6]. The PF_4 algorithm takes as input a parametric polynomial ideal and two monomial orderings on the variables and the parameters and returns a Gröbner system of the ideal with respect to a compatible elimination product of the given monomial orderings. We have implemented our new algorithm in Maple and give timings to compare its performance with those of (our implementation) of the Kapur et al. algorithm [16] and the DisPGB algorithm [21].

Keywords: Gröbner bases, Gröbner systems, F_4 algorithm, PF_4 algorithm, PGBMain algorithm, DisPGB algorithm.

2000 Mathematics subject classification: 13P10, 68W30.

*Corresponding Author

Received 07 September 2019; Accepted 28 December 2020

[DOI: 10.61186/ijmsi.19.1.117]

1. Introduction

Gröbner bases are a powerful tool and an important theoretical concept in the polynomial ring theory. The theory of Gröbner bases was introduced by Buchberger [4] in 1965, who named them after his supervisor Gröbner. *Buchberger's algorithm* is the first and most reputed method for computing Gröbner bases which was presented in Buchberger's Ph.D. thesis [2, 4]. Then, he proposed [3] two criteria to improve the performance of his algorithm. In [10] Gebauer and Möller presented an improvement of the Buchberger algorithm to apply efficiently Buchberger's criteria. In 1983, Lazard described an algorithm by using linear algebra methods [18] for the computation of Gröbner bases. Later on, Faugère proposed his two well-known algorithms, namely F_4 and F_5 for computing Gröbner bases (see [8, 9]). The main idea of the F_4 algorithm is the use of linear algebra techniques to perform simultaneously the reduction of a large number of critical pairs. This algorithm has been implemented in some of the computer algebra systems like Maple and Magma.

In this paper, we adapt the F_4 algorithm to compute *comprehensive Gröbner systems* (CGS's) for parametric polynomial ideals. For simplicity, throughout this paper we employ the terminology Gröbner system instead of CGS. Roughly speaking, Gröbner systems can be considered as an extension of Gröbner bases for polynomial ideals over fields to polynomial ideals with parametric coefficients. More precisely, a Gröbner system is a finite set of triples (so-called branches or segments); each branch contains a couple of null and non-null parametric sets (parametric constraints) and also a set of polynomials so that for every values of the parameters (specialization) one can find a branch so that the specialization satisfies its constraints, and the specialization of the corresponding polynomial set forms a Gröbner basis for the parametric ideal under the substitution of the values of the parameters. Gröbner systems have numerous applications in Mathematics and other field of sciences. In particular, we can point out algebraic geometry [11, 20, 21, 28], parametric linear algebra [6, 13], robotics [19, 21], automated geometry theorem proving [20, 22], automated geometry theorem discovery [22], electrical network [23] and so on. Since our study is focused on the theory of Gröbner systems, we review briefly this topic. The concept of *Gröbner system* was introduced by Weispfenning in [28]. He established also the existence of a Gröbner system for any given parametric polynomial ideal [28, Proposition 3.4 and Theorem 2.7] and presented the first algorithm to compute it [28, Theorem 3.6]. It should be noted that the solutions of parametric systems were studied at the same time using different methods by Kapur [15] and Sit [26], independently. In 2002, Montes [21] proposed a more efficient algorithm (DisPGB) for computing Gröbner systems (see also [7, 12]). Manubens and Montes in 2006, utilizing the concept

[DOI: 10.61186/ijmsi.19.1.117]

of discriminant ideals improved the DisPGB algorithm [19] and then introduced an algorithm to compute minimal canonical Gröbner systems [20]. In [27] Suzuki and Sato propounded an impressive improvement for computing Gröbner systems based on Kalkbrener's results [14] concerning specialization of parametric polynomial ideals and stability of the Gröbner bases under specialization. The Suzuki-Sato algorithm employed recursively computations of reduced Gröbner bases in an extension of the base polynomial ring. Afterward, Nabeshima proposed an improvement of this algorithm in [25] by reducing the number of branches generated by the Suzuki-Sato algorithm. Furthermore, Kapur et al. in 2010 proposed an efficient algorithm (PGBMain algorithm) for computing Gröbner systems by using combination of the Weispfenning [29] and the Suzuki-Sato algorithms, see [16, 17]. Finally, Montes and Wibmer in 2010 presented Gröbner Cover algorithm [24] based on a result for parametric polynomial systems proved by Wibmer in [30]. Gröbner Cover algorithm is the canonical algorithm for solving parametric polynomial systems and it plays a role similar to the reduced Gröbner basis for parametric systems. This algorithm computes a finite partition of parameter space into locally closed subsets together with certain polynomial data, from which the reduced Gröbner basis for each parameter point can be determined. The interested reader is referred to [22] for more details on theory of Göbner systems and their applications.

As mentioned above, PGBMain algorithm was presented by Kapur et al. in 2010. This algorithm at each iteration computes the Gröbner basis over a polynomial ring in terms of the variables and the parameters. Therefore, this step may be very expensive in practice, because the complexity of Gröbner basis computation is extremely influenced by the number of variables and degree of the given polynomials. Hence, it is important to design an efficient algorithm to reduces the computation in a polynomial ring in terms of only the variables. On the other hand, the DisPGB algorithm works in a polynomial ring in terms of only the variables and in addition benefits from almost all the improvements of the Buchberger algorithm which are applicable in the non-parametric setting. So, in this algorithm, like the conventional Buchberger's algorithm, a parametric S-polynomials is constructed and if its remainder is non-null then it is added to the basis set (see also [12]). In consequence, DisPGB creates new branches when a new polynomial with an undecidable coefficient is constructed and this may end with many number of branches which may turn the algorithm inefficient in practice. In order to tackle this problem, we present a parametric version of the F_4 algorithm. As the first step in this direction, we shall need a parametric linear algebra technique to achieve our purpose. It is worth noting that Gröbner systems may be not a powerful tool in solving any parametric polynomial system and that is why we already developed parametric linear algebra tools for studying parametric polynomial systems (in [6, Section 3], it is shown that applying parametric Gaussian elimination

[DOI: 10.61186/ijmsi.19.1.117]

method to solve a parametric polynomial system may be faster than applying Gröbner system method). Based on this approach using the basic ideas from the DisPGB structure [21], we present a parametric F_4 algorithm to compute Gröbner systems for parametric polynomial ideals. The proposed algorithm along with the algorithm due to Kapur et al [16] and the DiSPGB algorithm have been implemented in Maple and the comparison of their efficiency is discussed via a set of benchmark polynomials.

The rest of the paper is organized as follows. In Section 2, we review the basic definition and notations related to the theory of Gröbner systems. Then we present a parametric version of the F_4 algorithm in Section 3. Furthermore, in this section we illustrate the steps of this algorithm through a simple example. The efficiency of the proposed algorithm (compared to the PGBMain and DiSPGB algorithms) is discussed using several benchmark polynomials in Section 4.

2. Gröbner Systems

In this section, we review the basic definitions and notations that we use in the subsequent sections, for more details we refer the reader to [5, 22]. Throughout this paper, we consider $\mathcal{R} = \mathbb{K}[x_1, \ldots, x_n]$ the polynomial ring in terms of $x_1, \ldots, x_n$ over a field $\mathbb{K}$. Let $\mathcal{I} = \langle f_1, \ldots, f_k \rangle \subset \mathcal{R}$ be the polynomial ideal generated by the f_i's. We consider a monomial ordering $\prec$ on the set of all monomials (power products of the x_i's) of $\mathcal{R}$. For any $f \in R$, the *leading monomial* of f, denoted by $\mathrm{LM}_\prec(f)$, is the greatest monomial (with respect to $\prec$) appearing in f and its coefficient is the *leading coefficient* of f which denoted by $\mathrm{LC}_\prec(f)$. The *leading term* of f with respect to $\prec$ is the product $\mathrm{LT}_\prec(f) = \mathrm{LC}_\prec(f)\mathrm{LM}_\prec(f)$. The *leading monomial ideal* of $\mathcal{I}$ is defined to be $\mathrm{LM}_\prec(\mathcal{I}) = \langle \mathrm{LM}_\prec(f) \mid f \in \mathcal{I} \rangle$. A finite subset $\{g_1, \ldots, g_m\} \subset \mathcal{I}$ is called a *Gröbner basis* for $\mathcal{I}$ with respect to $\prec$ if $\mathrm{LM}_\prec(\mathcal{I}) = \langle \mathrm{LM}_\prec(g_1), \ldots, \mathrm{LM}_\prec(g_m) \rangle$. We refer e.g. to [5] for more details on the theory of Gröbner bases. Using these notations, we recall the definition of Gröbner systems for parametric polynomial ideals. For this purpose, let us consider $\mathcal{S} = \mathbb{K}[\mathbf{a}, \mathbf{x}]$ as a polynomial ring with parametric coefficients where $\mathbf{a} = a_1, \ldots, a_m$ is a sequence of parameters, $\mathbf{x} = x_1, \ldots, x_n$ is a sequence of variables and $\{\mathbf{x}\} \cap \{\mathbf{a}\} = \emptyset$. Thus a monomial $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ is denoted by x^α where $\alpha = (\alpha_1, \ldots, \alpha_n)$. Let $\prec_{\mathbf{x}}$ be a monomial ordering on the variables and $\prec_{\mathbf{a}}$ a monomial ordering on the parameters. For defining Gröbner systems, we shall need also to give recall a product ordering to specify an ordering on $\mathcal{S}$. The product of $\prec_{\mathbf{x}}$ and $\prec_{\mathbf{a}}$ denoted by $\prec_{\mathbf{x},\mathbf{a}}$, is defined as follows: For all $\alpha, \beta \in \mathbb{N}^n$ and $\gamma, \delta \in \mathbb{N}^m$, we write $\mathbf{x}^\alpha \mathbf{a}^\gamma \prec_{\mathbf{x},\mathbf{a}} \mathbf{x}^\beta \mathbf{a}^\delta$ if either $\mathbf{x}^\alpha \prec_{\mathbf{x}} \mathbf{x}^\beta$ or ($\mathbf{x}^\alpha = \mathbf{x}^\beta$and $\mathbf{a}^\gamma \prec_{\mathbf{a}} \mathbf{a}^\delta$).

In addition, if $\overline{\mathbb{K}}$ denotes the algebraic closure of $\mathbb{K}$ then from a specialization of parameters we mean a morphism

$$\sigma : \mathbb{K}[\mathbf{a}] \to \overline{\mathbb{K}}.$$

[DOI: 10.61186/ijmsi.19.1.117]

Therefore, for each f, we can write $\sigma(f) = f|_{\mathbf{a}=t_1,\ldots,t_m}$ where $\sigma(a_i) = t_i$. Furthermore, we say that a specialization σ satisfies $(N, W) \subset \mathbb{K}[\mathbf{a}] \times \mathbb{K}[\mathbf{a}]$ if $\sigma(p) = 0$ for all $p \in N$ and $\sigma(q) \neq 0$ for some $q \in W$. Equivalently, σ satisfies (N, W) if $(t_1, \ldots, t_m) \in \mathbb{V}(N) \setminus \mathbb{V}(W)$ where $\sigma(a_i) = t_i$. If $\mathbb{V}(N) \setminus \mathbb{V}(W) = \emptyset$ then (N, W) is said to be *inconsistent*. Also, N_i and W_i are called the null and non-null condition sets, respectively. The set of common zeros of $N \subset \mathcal{R}$ is denoted by $\mathbb{V}(N)$, for a set of polynomials N.

Definition 2.1. [27, Definition 1] Let $F \subset \mathcal{S}, G_i \subset \mathcal{S}$ and $(N_i, W_i) \subset \mathbb{K}[\mathbf{a}] \times \mathbb{K}[\mathbf{a}]$ for $i = 1, \ldots, \ell$. The triple set $\mathscr{G} = \{(N_i, W_i, G_i)\}_{i=1}^{\ell}$ is called a Gröbner system for $\langle F\rangle$ with respect to $\prec_{\mathbf{x},\mathbf{a}}$ over $\mathcal{V} \subseteq \overline{\mathbb{K}}^m$ if for any i we have

- $\sigma(G_i) \subset \overline{\mathbb{K}}[\mathbf{x}]$ is a Gröbner basis of $\langle\sigma(F)\rangle$ with respect to $\prec_{\mathbf{x}}$, for any specialization $\sigma : \mathbb{K}[\mathbf{a}] \to \overline{\mathbb{K}}$ satisfying (N_i, W_i)
- $\mathcal{V} \subseteq \bigcup_{i=1}^{\ell} \mathbb{V}(N_i) \setminus \mathbb{V}(W_i)$.

For each i, (N_i, W_i, G_i) is called a branch (segment) of the Gröbner system $\mathscr{G}$. Furthermore, if $\mathcal{V} = \overline{\mathbb{K}}^m$ then $\mathscr{G}$ is called a Gröbner system of F.

The concept of Gröbner system was introduced by Weispfenning in [28]. He proved that any parametric polynomial ideal has a Gröbner system and described an algorithm to compute it. Kapur et al. in [16] presented the efficient PGBMain algorithm for this computation. However, the output Gröbner system of this algorithm may contain several branches so that the corresponding Gröbner basis is $\{1\}$. On the other hand, by substituting only one branch instead of considering all these branches may reduce the consistency check and this can improve significantly the performance of the PGBMain algorithm, see [11] for more details. In the rest of the paper when we refer to the PGBmain algorithm we mean the modified version of this algorithm proposed in [11].

Example 2.2. Let $F = \{ay^3 + y^2x + 2, cy^2 + bz\} \subset \mathbb{K}[a, b, c, x, y, z]$ where x, y, z are variables and a, b, c are parameters. We consider the monomial orderings $z \prec_{drl} y \prec_{drl} x$ and $c \prec_{drl} b \prec_{drl} a$. Using our implementation of the PGBMain algorithm, we can compute a Gröbner system for $\langle F\rangle$ as follows

$$\begin{cases} ([\,], & [bc], & [abyz + bxz - 2c, cy^2 + bz]) \\ ([b, c], & [\,], & [ay^3 + y^2x + 2]) \\ ([c], & [b], & [bz, ay^3 + y^2x + 2])) \\ ([b], & [c], & [1])). \end{cases}$$

For instance, if we set $a = 1, b = 2$ and $c = 3$ then the first branch corresponds to these values of parameters and so $\{zx+zy-3, 3y^2+2z\}$ is a Gröbner basis for the ideal $\langle F\rangle\,|_{a=1,b=2,c=3}$.

3. Description of a Parametric F$_4$ Algorithm

Faugère's F4 algorithm [8] has a structure similar to that of the Buchberger algorithm, however in contrast to it which computes S-polynomial remainders

[DOI: 10.61186/ijmsi.19.1.117]

one by one, several S-polynomial remainders are performed simultaneously via row-reduction on a suitable matrix (generally *sparse matrix*) to do the reductions in parallel. This structure, along with the use of linear algebra methods, is the cornerstone of the F_4 algorithm. We refer e.g. to [5, 8] for more details on the structure of the F_4 algorithm. In this section, we present a parametric F_4 algorithm which can be considered as a generalization of the F_4 algorithm to polynomial ideals with parametric coefficients. We remark that our generalization is non-trivial in the sense that, in several stages in the F_4 algorithm we shall perform linear and non-linear reductions, and it is non-trivial to handle the parametric variants of all these reductions. In this direction, we apply the GES algorithm [6] with slight modifications. This algorithm computes a Gaussian elimination system for a parametric matrix (equivalently a parametric linear system corresponding to the input matrix). However, we apply this algorithm on non-linear polynomials to make a linear inter-reduction, and we look for their Gaussian forms according to parametric constraints. To this end, we shall linearize the input polynomials by replacing each monomial appearing in the polynomials by new variables. The engine of the GES algorithm is the LDS algorithm [6] which discusses the dependency of a linear parametric polynomial with respect to a given set of parametric polynomials without the use of Gröbner systems. For the convenience of the reader, we review shortly the LDS algorithm from [6]. Below, we let Sys be a variable which is initialized to empty set, and finally it is the output linear dependency system. In addition, the LDS algorithm receives as input (N, W, F, f) where (N, W) is a pair of condition sets, F is a set of linear parametric polynomials (which forms a parametric Gröbner basis with respect to the given condition sets) and f is a linear parametric polynomial and returns a finite set of triples of the form $(N_1, W_1, [flag, Q, g])$ where (N_1, W_1) is a pair of condition sets, flag is a Boolean variable, Q represents the quotients of the division and g is the normal form of f with respect to F. If flag is true then $g = 0$ and in consequence f is linear dependent on F with respect to (N_1, W_1), and if it is false then f is linear independent modulo F with respect to (N_1, W_1). We use below the function NormalForm which receives as input a polynomial p, a Gröbner basis $G = \{g_1, \ldots, g_m\}$ and a monomial ordering $\prec$ and returns f and $Q = [q_1, \ldots, q_m]$ where f is the normal form of p by G and $p = q_1 g_1 + \cdots + q_m g_m + f$. Finally, we shall mention that the LDS algorithm uses an arbitrary monomial ordering on the variables appearing in F and f to discuss the parametric linear dependency of f on F and this does not change the correctness of the output.

[DOI: 10.61186/ijmsi.19.1.117]

Algorithm 1 LDS (Linear Dependency System)

Require: $G \subset \mathcal{S}$; a linear set which is a reduced Gröbner basis w.r.t. the product of the monomial orderings $\prec_{\mathbf{x}}$ and $\prec_{\mathbf{a}}$ provided that a conditions pair (N, W) is satisfied and $g \in \mathcal{S}$; a parametric linear polynomial

Ensure: A linear dependency system of g on (N, W, G)

Sys:= {}
f, Q :=NormalForm(g,GröbnerBasis($N, \prec_{\mathbf{a}}$), $\prec_{\mathbf{a}}$)
f', Q' :=NormalForm($f, G, \prec_{\mathbf{x}}$)
if $f' = 0$ **then**
  Sys:=Sys $\bigcup\{(N, W, [true, Q', 0])\}$
else
  $A := \{a_{i_1}, \ldots, a_{i_t}\}$ where $f' = a_{i_1}x_{i_1} + \cdots + a_{i_t}x_{i_t}$ with $a_{i_j} \neq 0$ and $x_{i_1} \succ_{\mathbf{x}} \cdots \succ_{\mathbf{x}} x_{i_t}$
  for j **from** 1 **to** t **do**
    if a_{i_j} is not constant **then**
      Sys:=Sys $\bigcup\{(N \cup \{a_{i_1}, \ldots, a_{i_{j-1}}\}, W \cup \{a_{i_j}\}, [false, Q', f'|_{a_{i_1}=0,\ldots,a_{i_{j-1}}=0}])\}$
    else
      Sys:=Sys $\bigcup\{(N \cup \{a_{i_1}, \ldots, a_{i_{j-1}}\}, W, [false, Q', f'|_{a_{i_1}=0,\ldots,a_{i_{j-1}}=0}])\}$
      Return(Sys)
    end if
  end for
  Sys:=Sys $\bigcup\{(N \bigcup A, W, [true, Q', 0])\}$
end if
Return(Sys)

The behavior of the above algorithm is illustrate by a simple example.

Example 3.1. Consider $(N, W, G) = ([\,], [a-1, b-1, c], [x + av, y + bu, z])$ and $g = (a-2)x + ty + cz + du + (3-b)v$. We fix the monomial orderings $t \prec_{lex} d \prec_{lex} c \prec_{lex} b \prec_{lex} a$ and $v \prec_{lex} u \prec_{lex} z \prec_{lex} y \prec_{lex} x$ on the parameters and the variables, respectively. At the beginning, we set

$$\begin{array}{rcl} f, Q & := & \text{NormalForm}(g, N, \prec_{\mathbf{a}}) = (a-2)x + ty + cz + du + (3-b)v, [\,] \\ f', Q' & := & \text{NormalForm}(f, G, \prec_{\mathbf{x}}) = (-bt + d)u + (-a^2 + 2a + 3 - b)v, [a-2, t, c]. \end{array}$$

Since $f' \neq 0$ we consider the set $A = \{-bt + d, -a^2 + 2a - b + 3\}$ of the coefficients of f'. By the structure of the algorithm, we consider first two pairs $([\,], [a-1, b-1, c, -bt + d])$ and $([-bt + d], [a-1, b-1, c, -a^2 + 2a - b + 3])$ which are consistent and therefore we have

$$\text{Sys} = \left\{ \begin{array}{l} ([\,], [a-1, b-1, c, -bt + d], [false, [a-2, t, c], (-bt + d)u + (-a^2 + 2a + 3 - b)v]), \\ ([-bt + d], [a-1, b-1, c, -a^2 + 2a - b + 3], [false, [a-2, t, c], (-a^2 + 2a + 3 - b)v]). \end{array} \right.$$

Since A does not contain any constant and the pair $(N \cup \{A\}, W)$ is consistent then a linear dependency system of g on (N, W, G) is as follow:

$$\text{Sys} = \left\{ \begin{array}{l} ([\,], [a-1, b-1, c, -bt + d], [false, [a-2, t, c], (-bt + d)u + (-a^2 + 2a + 3 - b)v]), \\ ([-bt + d], [a-1, b-1, c, -a^2 + 2a - b + 3], [false, [a-2, t, c], (-a^2 + 2a + 3 - b)v]), \\ ([-bt + d, -a^2 + 2a - b + 3], [a-1, b-1, c], [true, [a-2, t, c], 0]). \end{array} \right.$$

Using this algorithm, we are willing to present an efficient algorithm to compute a Gaussian elimination system for a set of (non necessary linear) parametric polynomials. Below, the notion Sys stands for a variable which

[DOI: 10.61186/ijmsi.19.1.117]

is initialized to empty set, and finally it is the output Gaussian elimination system. We note that each recorded branch in Sys contains a triple (N, W, G) where (N, W) is a pair of condition sets and G is a Gaussian elimination form of the input parametric polynomials set with respect to (N, W). In the following algorithm $Y[i]$ denotes i-th element of a list or set Y.

Algorithm 2 GES (Gaussian Elimination System)

Require: $N \subset \mathbb{K}[\mathbf{a}]$; null condition set, $W \subset \mathbb{K}[\mathbf{a}]$; non-null condition set, $F \subset \mathbb{K}[\mathbf{a}, \mathbf{x}]$; a parametric polynomial set
Ensure: A Gaussian elimination system of F according to N and W
Sys:= {}
$M := Mon(F) = [m_1, \ldots, m_t]$ (the set of all monomials in terms of the x_i's appearing in F)
$[Y_1, \ldots, Y_t]$:=A list of tag variables corresponding to $Mon(F)$
$L := \phi(F)$ where ϕ is a linear map sending each m_i into Y_i
$A := \{(N, W, \{\}, L[1], L)\}$
while $A \neq \{\}$ **do**
  $a := A[1]$ and $A := A \setminus \{a\}$
  if $a[5] = \{\}$ **then**
    $G := \phi^{-1}(a[3])$
    Sys:=Sys $\bigcup\{(a[1], a[2], G)\}$
  else
    $G := a[5] \setminus \{a[4]\}$
    $g := G[1]$
    P :=LDS$(a[1], a[2], a[3], a[4])$
    for i **from** 1 **to** $|P|$ **do**
      Let $P[i] = (N_1, W_1, [flag, Q, f])$
      if $flag = true$ **then**
        $A := A \cup \{(N_1, W_1, a[3], g, G)\}$
      else
        $A := A \cup \{(N_1, W_1, a[3] \cup \{f\}, g, G)\}$
      end if
    end for
  end if
end while
Return (Sys)

Theorem 3.2. *The GES algorithm terminates in finitely many steps and is correct.*

Proof. Since F is finite and for any $f \in F$ the LDS algorithm computes the linear dependency system of f in finitely many steps then the GES algorithm trivially terminates in finitely many steps. Also, the correctness of the LDS algorithm guarantees the correctness of this algorithm. More precisely, by the structure of the algorithm, we discuss a new polynomial $f \in F$ using the LDS algorithm. If it is linear dependent on the computed basis, then it is removed. Otherwise, its normal form with respect to the computed basis is added into

[DOI: 10.61186/ijmsi.19.1.117]

the basis. Thus, each branch contains a Gaussian elimination form of the input parametric polynomial set with respect to the corresponding conditions pair. □

Example 3.3. Let us consider $F = \{ax^2+by+1, cz^3+(a-1)y-b, (a-b)y^2+(c-1)xy-2\} \subset \mathbb{K}[a,b,c][x,y,z]$ as a set of polynomials with parameters. Using the GES algorithm, we get the following Gaussian elimination system for F when $(N,W) = ([a-1],[c])$.

$$\begin{cases} ([a-1],[c,c-1], & [x^2+by+1, cz^3-b, (1-b)y^2+cxy-xy-2]),\\ ([c-1,a-1],[b-1], & [x^2+by+1, z^3-b, (1-b)y^2-2]),\\ ([c-1,b-1,a-1],[\,], & [x^2+y+1, z^3-1, -2]). \end{cases}$$

We deal now with presenting a parametric F_4 algorithm, so-called PF_4 to compute Gröbner systems for parametric polynomial ideals. This algorithm, which is a generalization of F_4 algorithm to parametric coefficients, receives as input a parametric polynomial set F and two monomial orderings on variables and parameters and outputs a Gröbner system for the ideal generated by F. In this algorithm, we use a global variable OUTSYS which is initially the empty sequence and at each iteration of two algorithms PF_4 and PF_4Basis, some new segments are added to this sequence. At the end, OUTSYS is a Gröbner system of the input parametric polynomial ideal.

Algorithm 3 PF_4

Require: $F \subset \mathbb{K}[\mathbf{a},\mathbf{x}] = \mathbb{K}[a_1,\ldots,a_m,x_1,\ldots,x_n]$, $\prec_{\mathbf{x}}, \prec_{\mathbf{a}}$; two monomial orderings
Ensure: G; A Gröbner system of $\langle F\rangle$ with respect to $\prec_{\mathbf{x},\mathbf{a}}$

```
OUTSYS := NULL
A :=GES([ ], [ ], F)
for (N_n, W_n, F_n) ∈ A do
  if F_n = [ ] then
    OUTSYS:= OUTSYS, (N_n, W_n, [ ])
  end if
  if there is any constant or non-zero parameter in F_n then
    OUTSYS:= OUTSYS, (N_n, W_n, [1])
  else
    t := |F_n| (the cardinality of F_n)
    B := [[{i,j}, deg(lcm(LM_{≺x}(F_n[i]), LM_{≺x}(F_n[j])))] | 1 ≤ i < j ≤
    t, gcd(LM_{≺x}(F_n[i]), LM_{≺x}(F_n[j])) ≠ 1]
    if B = [ ] then
      OUTSYS:= OUTSYS, (N_n, W_n, F_n)
    else
      SYS := [[N_n, W_n, F_n, t, B]]
    end if
  end if
  PF4Basis(SYS)
end for
```


[DOI: 10.61186/ijmsi.19.1.117]

We continue this section with the description of the PF_4Basis algorithm which is the most important part of the PF_4 algorithm. Below, we used the ComputeM sub-algorithm proposed by Faugère in [8] and its simple version was presented in [5, page 571]. This sub-algorithm receives two sets of polynomials L and G and its goal is to produce a polynomial set H so that the following conditions hold:

i) $L \subseteq H$,
ii) if there exists $f_\ell \in G$ with $\mathrm{LM}(f_\ell)|x^\beta$ for some monomial x^β appearing in some element in H then $x^\alpha f_\ell$ is included in H where $\mathrm{LM}(x^\alpha f_\ell) = x^\beta$.

So, it is clear that this sub-algorithm can construct H within a finite loop. Furthermore, in the next algorithm, the Newpolys is a procedure which gets two lists of polynomials F, G and returns the list of those polynomials $f \in F$ so that $\mathrm{LM}(f) \notin \langle \mathrm{LM}(G) \rangle$. Also, NormalSet$(F, G, \prec)$ computes the normal form of all polynomials of F modulo G with respect to a monomial ordering $\prec$, namely NormalSet$(F, G, \prec) = \{\overline{f}^G_\prec | f \in F\}$ where $\overline{f}^G_\prec$ is a remainder of f on division by G with respect to $\prec$. The algorithm maintains a list B of pairs whose corresponding S-polynomials have not be reduced. But it is worth noting that B is now a set of not ordered pairs. For this purpose, the degree of a pair $\{i, j\}$ is defined to be $\deg(\mathrm{lcm}(\mathrm{LM}_{\prec_x}(f_i), \mathrm{LM}_{\prec_x}(f_j)))$ and then one can use the *degree-normal selection strategy* to order the pairs. Like the original form of the F_4 algorithm, the degree-normal selection strategy consists of choosing the set of all pairs of the minimal degree in each step.

It should be mentioned that the PF_4 algorithm has a structure similar to the Buchberger algorithm in which we consider the set of critical pairs to study and therefore applying Buchberger's criteria may be helpful to skip some unnecessary critical pairs. In this direction, the PF_4Basis algorithm benefits from the Update algorithm exhibited in [1, page 230] to discard apriori superfluous critical pairs by employing Buchberger's criteria (this algorithm is due to Gebauer-Möller [10], and it has been presented in a more clear manner in [1, page 230]). Moreover, in order to enhance the efficiency of the PF_4Basis algorithm, we keep track of the computations as follows: Each branch $sys \in SYS$ in the PF_4Basis algorithm is of the form $sys = (a[1], \ldots, a[5])$ containing the following information:

- $a[1]$: The null condition set
- $a[2]$: The non-null condition set
- $a[3]$: The set of polynomials which forms a Gröbner basis for $\langle F \rangle$ with respect to $\prec_{\mathbf{x}}$
- $a[4]$: The cardinality of $a[3]$
- $a[5]$: A list of pairs so that the first component each element is a pair $\{i, j\}$ and second component is $\deg(\mathrm{lcm}(\mathrm{LM}_{\prec_x}(a[3][i]), \mathrm{LM}_{\prec_x}(a[3][j])))$.

If $L = [f_1, \ldots, f_\ell]$ is a list, then sequence $f_1, \ldots, d_\ell$ is denoted by $op(L)$.

[DOI: 10.61186/ijmsi.19.1.117]

Algorithm 4 PF$_4$Basis

Require: $N \subset \mathbb{K}[\mathbf{a}]$; null condition set, $W \subset \mathbb{K}[\mathbf{a}]$; non-null condition set, $F \subset \mathbb{K}[\mathbf{a}, \mathbf{x}]$; t; the cardinality of F and $Cpairs$; a list of pairs so that the first component is a pair of integers $\{i, j\}$ and second component is $\deg(\mathrm{lcm}_{\prec_{\mathbf{x}}}(\mathrm{LM}(F[i]), \mathrm{LM}_{\prec_{\mathbf{x}}}(F[j])))$

Ensure: Decomposing the space of parameters into a finite set of parametric cells and for each cell associating a finite set of parametric polynomials

```
B := Cpairs
while SYS ≠ [ ] do
  sys := SYS[1] and remove it from SYS
  G := sys[3]
  B := sys[5]
  Select Bp ⊆ B using degree-normal selection strategy
  B := B \ Bp
  Bsys := B
  L := {lcm(LM(f_i),LM(f_j))/LM(f_i) . f_i, lcm(LM(f_i),LM(f_j))/LM(f_j) . f_j | {i,j} ∈ Bp}
  H :=ComputeM(L, G)
  Ges :=GES(sys[1], sys[2], H)
  allNPi := [Newpolys(Ges[i][3], H), i = 1, ..., | Ges |]
  for j from 1 to | allNPi | do
    t := sys[4]
    G := NormalSet(sys[3], Ges[j][1], ≺_a);
    B := Bsys
    for ℓ from 1 to | allNPi[j] | do
      t := t + 1
      B :=Update(allNPi[j][ℓ], G, B)
      G := [op(G), allNPi[j][ℓ]]
      if B = [ ] then
        OUTSYS:= OUTSYS, (Ges[j][1], Ges[j][2], G)
      else
        SYS := [op(SYS), [Ges[j][1], Ges[j][2], G, t, B]]
      end if
    end for
  end for
  {PF4Basis(SYS[m])}_{m=1}^{|SYS|}
end while
Return (OUTSYS)
```

Theorem 3.4. *The* PF$_4$ *algorithm terminates in finitely many steps and is correct.*

Proof. The termination of this algorithm is essentially ensured by those of the GES and F_4 algorithms. More precisely, we can consider this computation like a tree graph and each node corresponds to a triple which is the output of the GES algorithm. The number of branches is finite (due to the termination of the original F_4 algorithm). Moreover, the number of nodes in this tree is finite (by termination of the GES algorithm) and all these arguments together conclude the termination of the algorithm.

[DOI: 10.61186/ijmsi.19.1.117]

Also, the correctness of the algorithm is guaranteed by the correctness of the GES and the F_4 algorithms. Indeed, by the structure of the algorithm, at each step we have a condition sets $(sys[1], sys[2])$, a set of polynomials G and a list of critical pairs B. Then, we choose the critical pairs with lowest possible degree and start discussing them using the GES algorithm. We obtain in turn new polynomials by decomposing $(sys[1], sys[2])$. For each new polynomial with its own condition sets, we add it into G and also update B. Since, we basically follow the structure of the F_4 algorithm, we get at the end a Gröbner system of the input ideal. □

We illustrate the steps of the PF_4 algorithm by the following simple example.

EXAMPLE 3.5. Let $F = [(c^2-1)x^2+b^2-1, (a^2-1)xy^2+c+b] \subset \mathbb{K}[a,b,c][x,y]$ where x, y are varibles and a, b, c are parameters. We consider the monomial orderings $y \prec_{lex} x$ and $c \prec_{lex} b \prec_{lex} a$. We want to compute a Gröbner system of the ideal generated by F using the PF_4 algorithm. At the beginning, the GES algorithm is called to compute a Gaussian elimination system of F and the output of this algorithm is as follows

$$A = \begin{cases} ([\,], [a-1, a+1, c-1, c+1], & [c^2x^2+b^2-x^2-1, a^2xy^2-xy^2+b+c]), \\ ([a^2-1], [b+c, c-1, c+1], & [c^2x^2+b^2-x^2-1, b+c]), \\ ([b+c, a^2-1], [c-1, c+1], & [c^2x^2+c^2-x^2-1]), \\ ([c^2-1], [a^2-1, b-1, b+1], & [b^2-1, a^2xy^2-xy^2]), \\ ([c^2-1, a^2-1], [b-1, b+1], & [b^2-1]), \\ ([c^2-1, b^2-1], [a-1, a+1], & [a^2xy^2-xy^2+b+c]), \\ ([c^2-1, b^2-1, a^2-1], [b+c], & [b+c]), \\ ([c^2-1, b+c, a^2-1], [\,], & [\,]). \end{cases}$$

A has eight segments and as it is seen the last Gröbner basis is $[\,]$. Furthermore, the associated Gröbner bases of the second, fourth, fifth and seventh branches are all $[1]$. Thus, the corresponding branches are added in advance to OUTSYS:

$$\mathrm{OUTSYS} = \begin{cases} ([a^2-1], & [b+c, c-1, c+1], & [1]), \\ ([c^2-1], & [a-1, a+1, b-1, b+1], & [1]), \\ ([c^2-1, a^2-1], & [b-1, b+1], & [1]), \\ ([c^2-1, b^2-1, a^2-1], & [b+c], & [1]), \\ ([c^2-1, b+c, a^2-1], & [\,], & [\,]). \end{cases}$$

Now for simplicity, we select the first triple of A namely;

$$([\,], [a-1, a+1, c-1, c+1], [c^2x^2+b^2-x^2-1, a^2xy^2-xy^2+b+c])$$

and ignore the other branches. Therefore, we set SYS as the following

$$[[\,], [a-1, a+1, c-1, c+1], [c^2x^2+b^2-x^2-1, a^2xy^2-xy^2+b+c], 2, [[[1,2],4]]].$$

Now, PF_4BASIS is called and receives a quintuple of SYS. Let sys be the only element of SYS. In the `while` loop, we have

$$L = H = \{\frac{x(a^2xy^2-xy^2+b+c)}{a^2-1}, \frac{y^2(c^2x^2+b^2-x^2-1)}{c^2-1}\}.$$

[DOI: 10.61186/ijmsi.19.1.117]

Now, the GES algorithm receives as input $sys[1], sys[2], H$ and gives a Gaussian elimination system of H as follows

$$Ges = \begin{cases} ([\,], [b+c, a-1, a+1, c-1, c+1], [a^2x^2y^2 - x^2y^2 + bx + cx, a^2b^2y^2 + y^2 \\ \qquad\qquad -a^2y^2 - b^2y^2 - bc^2x - c^3x + bx + cx]), \\ ([b+c], [a-1, a+1, c-1, c+1], [a^2x^2y^2 - x^2y^2, (a^2c^2 - a^2 - c^2 + 1)y^2]). \end{cases}$$

Since, Ges has two branches so NEWPOLYS function is called twice and so we can write

$$\text{NEWPOLYS}(Ges[1][3], H) = [a^2b^2y^2 - a^2y^2 - b^2y^2 - bc^2x - c^3x + bx + cx + y^2]$$

$$\text{NEWPOLYS}(Ges[2][3], H) = [a^2c^2y^2 - a^2y^2 - c^2y^2 + y^2].$$

Notice that NEWPOLYS is a procedure which receives two lists of polynomials F, G and returns those polynomials $f \in F$ so that $\text{LM}(f) \notin \langle \text{LM}(G) \rangle$. Hence, in this step, the list of all new polynomials added to $allNPi$ is

$$allNPi = [a^2b^2y^2 - a^2y^2 - b^2y^2 - bc^2x - c^3x + bx + cx + y^2], [a^2c^2y^2 - a^2y^2 - c^2y^2 + y^2].$$

The cardinality of $allNPi$ is 2. So after passing two `for` loop and applying the UPDATE algorithm, one gets

$$t = 3, G = [c^2x^2 + c^2 - x^2 - 1, a^2xy^2 - xy^2, a^2c^2y^2 - a^2y^2 - c^2y^2 + y^2], B = [[\{2, 3\}, 3]]$$

Note that each element of B contains the indices corresponding to a critical pair along with the degree of the corresponding S-polynomial. Therefore, SYS is enlarged by two new quintuples as the following

$$SYS = \begin{cases} ([\,], [a-1, a+1, b+c, c-1, c+1], & [c^2x^2 + b^2 - x^2 - 1, a^2xy^2 - xy^2 + b + c, \\ & a^2b^2y^2 - a^2y^2 - b^2y^2 - bc^2x - c^3x + \\ & bx + cx + y^2], \\ & 3, \quad [[\{1, 3\}, 2], [\{2, 3\}, 3]]), \\ ([b+c], [a-1, a+1, c-1, c+1], & [c^2x^2 + c^2 - x^2 - 1, a^2xy^2 - xy^2, \\ & a^2c^2y^2 - a^2y^2 - c^2y^2 + y^2], \\ & 3, \quad [[\{2, 3\}, 3]]). \end{cases}$$

Since SYS has two segments the computation is continued into two branches and again for simplicity, we focus on the second member and ignore the first one. The PF_4BASIS algorithm is called again and receives as input the following data

$$[b+c], [a-1, a+1, c-1, c+1], [c^2x^2 + c^2 - x^2 - 1, a^2xy^2 - xy^2, (a^2c^2 - a^2 - c^2 + 1)y^2], 3, [[\{2, 3\}, 3]].$$

Since SYS is not empty, so we enter into the `while` loop by setting $L = H = \{xy^2\}$. The Gaussian elimination system of H is

$$Ges = [[[b+c], [a-1, a+1, c-1, c+1], [xy^2]]].$$

On the other hand, $Ges[1][3] = H = [xy^2]$. Accordingly,
$\text{NEWPOLYS}([xy^2], [xy^2]) = [\,]$ which deduces that $allNPi = [\,]$ and we have

$$t = 3, G = [c^2x^2 + c^2 - x^2 - 1, a^2xy^2 - xy^2, a^2c^2y^2 - a^2y^2 - c^2y^2 + y^2], B = [\,].$$

[DOI: 10.61186/ijmsi.19.1.117]

Since, $B = [\,]$ so the sixth triple of Gröbner system of $\langle F\rangle$ added to OUTSYS is

$$[[b{+}c], [a{-}1, a{+}1, c{-}1, c{+}1], [c^2x^2{+}c^2{-}x^2{-}1, a^2xy^2{-}xy^2, (a^2c^2{-}a^2{-}c^2{+}1)y^2]].$$

If we consider all the ignored branches during the calculation then we obtain the following Gröbner system of $\langle F\rangle$ saved in OUTSYS:

$$\begin{cases}
([a^2-1], [b+c, c-1, c+1], & [1]),\\
([c^2-1], [a-1, a+1, b-1, b+1], & [1]),\\
([c^2-1, a^2-1], [b-1, b+1], & [1]),\\
([c^2-1, b^2-1, a^2-1], [b+c], & [1]),\\
([c^2-1, b+c, a^2-1], [\,], & [\,]),\\
\\
([b^2-1], [a-1, a+1, b+c, c-1, c+1, 2bc+c^2+1], & [c^2x^2-x^2, a^2xy^2-xy^2+b+c, bc^2x\\
& +c^3x-bx-cx, 2bc^3+c^4-2bc-1]),\\
\\
([b+c, a^2-1], [c-1, c+1], & [c^2x^2+c^2-x^2-1]),\\
([c^2-1, b^2-1], [a-1, a+1], & [a^2xy^2-xy^2+b+c]),\\
\\
([b+c], [a-1, a+1, c-1, c+1], & [c^2x^2+c^2-x^2-1, a^2xy^2-xy^2,\\
& a^2c^2y^2-a^2y^2-c^2y^2+y^2]),\\
\\
([\,], [a-1, a+1, b-1, b+1, b+c, c-1, c+1], & [c^2x^2+b^2-x^2-1, a^2xy^2-xy^2+b\\
& +c, (a^2b^2-a^2-b^2)y^2-bc^2x-c^3x\\
& +bx+cx+y^2, (a^4b^2-a^4-2a^2b^2)y^4\\
& +2a^2y^4+b^2c^2+(y^4-1)b^2+2bc^3-\\
& 2bc+c^4-c^2-y^4]).
\end{cases}$$

4. **Experiments and Results**

In this section, we aim to compare the performance of the PF_4 algorithm with PGBMain and DisPGB algorithms. For this purpose, we have implemented all the algorithms described in this paper in Maple 15. Below, we refer to the Kapur et al. algorithm as "PGBMain") and to the Montes DisPGB algorithm as Improved-DisPGB which involves an improvement of the DisPGB algorithm [21] by installing Update algorithm [12]. In this direction, the following parametric ideals have been chosen in the ring $\mathcal{S} = \mathbb{Q}[a, b, c, d, m, n, r, t][x, y, z, u, v, w]$, and we aimed to compute a Gröbner system of the ideal generated by each list of polynomials with respect to the product of the orderings $v \prec_{lex} w \prec_{lex} u \prec_{lex} z \prec_{lex} y \prec_{lex} x$ and $t \prec_{lex} r \prec_{lex} n \prec_{lex} m \prec_{lex} d \prec_{lex} c \prec_{lex} b \prec_{lex} a$.

- EX.1= $[ab^4cuxz - a - c, aby^2 - a^2 + b^2, abuxz - c]$
- EX.2= $[(a-c)xz - x, (-b^3 + a^2)uxz - ab, (a+b)y - a^2]$
- EX.3= $[(c^2-1)x^2y + b^2 - 1, (a^2-1)x^2z + c + b, (a-b)y^2z - x - 1, bxy + a - c]$
- EX.4= $[bx^2z^3 - n^3 + n, cx^2y^3 - a^3 - a, dx^3y^2 - m^3 - m]$
- EX.5= $[(c-1)x^3 + a^2b - c, (1-b)zy^4x + b + a, (1-c)yzx - a - b - c]$
- EX.6= $[abcxyz - a - b - c, abxy - a - b, ax^3 - bc, by^3 - c, cz^3 - a]$
- EX.7= $[(bc{-}1)x^2{+}c^2{-}a, (ab{-}1)z^2{-}c{+}a, y^2{-}(b{-}1)x{-}1, (b{+}c{+}a)z^2{+}a{+}b{+}c]$
- EX.8= $[(c{-}a{-}b)x^3z^3{+}c^3{-}a{-}b, (b{-}a{-}c)z^2y^5{-}c{-}a{-}m, (a{-}m{+}n)z^2{-}a{+}b]$
- EX.9= $[(a-1)xyz + a, (b-2)y^2 + ab, (c+a)xy - a - 1]$
- EX.10= $[b^4x^2 + ac^3 - c, aby^3 + b^6 - a, bcz^5 - 3ac + 1]$

[DOI: 10.61186/ijmsi.19.1.117]

- EX.11= $[ab^4tux^3 - x - a^3 - n, abxy^3 + b^4 - a^2 + a, anxz^3 + a - 1]$
- EX.12= $[(m^5 - b)x + n - 1, y + (b^3m^2 - n)z - 1, b^3z^2 - mz - 1]$
- EX.13= $[rx^5 + (ab - c)z - n, cy^3 + acx + dn, z^3 - (c - t)y]$
- EX.14= $[abxy + ay^3 - cz + 1, ax^2 + ax + cu, tu^3 + tu, bz^3 + mnx - bz]$

Example	Method	Time (sec.)	Used Memory (GB)
EX.1	PF4	0.38	0.007
	PGBMain	0.51	0.016
	FirstGB	0.2	0.006
	Improved-DisPGB	0.54	0.018
EX.2	PF4	0.9	0.02
	PGBMain	0.41	0.012
	FirstGB	0.3	0.005
	Improved-DisPGB	1.44	0.035
EX.3	PF4	11.64	0.72
	PGBMain	—	—
	FirstGB	—	—
	Improved-DisPGB	—	—
EX.4	PF4	3.52	0.09
	PGBMain	27.81	1.91
	FirstGB	0.32	0.015
	Improved-DisPGB	6.1	0.23
EX.5	PF4	24.26	1.84
	PGBMain	—	—
	FirstGB	8.41	0.82
	Improved-DisPGB	17.93	1.99
EX.6	PF4	12.48	1.2
	PGBMain	—	—
	FirstGB	—	—
	Improved-DisPGB	—	—
EX.7	PF4	1.13	0.041
	PGBMain	2.17	0.1
	FirstGB	0.02	0.016
	Improved-DisPGB	1.84	0.048
EX.8	PF4	1.43	0.036
	PGBMain	—	—
	FirstGB	197.25	27.64
	Improved-DisPGB	2.59	0.075
EX.9	PF4	0.67	0.022
	PGBMain	031	0.007
	FirstGB	0.14	0.001
	Improved-DisPGB	1.04	0.029
EX.10	PF4	0.21	0.005
	PGBMain	1.89	0.16
	FirstGB	1.41	0.12
	Improved-DisPGB	0.32	0.01
EX.11	PF4	9.71	0.45
	PGBMain	11.23	0.85
	FirstGB	4.81	0.51
	Improved-DisPGB	50.21	6.3
EX.12	PF4	0.27	0.01
	PGBMain	56.11	5.62
	FirstGB	40.54	5.19
	Improved-DisPGB	0.31	0.014
EX.13	PF4	65.87	3.41
	PGBMain	—	—
	FirstGB	15.12	1.12
	Improved-DisPGB	3.21	0.45
EX.14	PF4	265.13	10.78
	PGBMain	523.17	57.91
	FirstGB	59.23	7.98
	Improved-DisPGB	2.41	0.32

The results are shown in the above tables where the third and fourth columns show respectively the CPU time (in seconds) and the amount of used memory (in gigabytes) of the total computation by the corresponding method. Furthermore, the row "First GB" stands for the computation of the reduced Gröbner

[DOI: 10.61186/ijmsi.19.1.117]

basis of the corresponding ideal in the polynomial ring $\mathbb{K}[\mathbf{a}, \mathbf{x}]$ with respect to $\prec_{\mathbf{x},\mathbf{a}}$ using the MAPLE function `Basis`. It is worth noting that, this computation is needed the first step in the PGBMAIN algorithm to compute a Gröbner system with respect to $\prec_{\mathbf{x},\mathbf{a}}$. Also, the symbol "—" means that the results can not computed within 600 seconds. The timings were conducted on personal computer with 5 core, 4 GB RAM and 64 bits under the Windows 10 operating system.

ACKNOWLEDGEMENT

The authors would like to thanks anonymous reviewers for their helpful comments. This work was partially supported by IPM.

REFERENCES

1. T. Becker, V. Weispfenning, *Gröbner Bases: a Computational Approach to Commutative Algebra*, New York: Springer-Verlag, 1993.
2. B. Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassen Ringes nach einem Nulldimensionalen Polynomideal*, Innsbruck: Univ. Innsbruck, Mathematisches Institut (Diss.), 1965.
3. B. Buchberger, A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Bases, *Symbolic and algebraic computation, EUROSAM '79, int. Symp., Marseille 1979, Lect. Notes Comput. Sci.*, **72**, (1979), 3-21.
4. B. Buchberger, Bruno Buchberger's Ph.D. Thesis 1965: An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal Translation from the German, *J. Symb. Comput.*, **41**(3-4), (2006), 475-511.
5. D. Cox, A. Little, D. O'Shea *Ideals, Varieties, and Algorithms. An introduction to Computational Algebraic Geometry and Commutative Algebra*, 4th edition, Springer, 2015.
6. M. Dehghani Darmian, A. Hashemi, Parametric FGLM Algorithm, *J. Symb. Comput.*, **82**, (2017), 38-56.
7. M. Dehghani Darmian, A. Hashemi, A. Montes, Erratum to "A new algorithm for discussing Gröbner bases with parameters" [J. Symbolic Comput. 33 (1-2) (2002) 183-208], *J. Symb. Comput.*, **46**(10), (2011), 1187-1188.
8. J.-C. Faugère, A New Efficient Algorithm for Computing Gröbner Bases (F4), *J. Pure Appl. Algebra*, **139**(1-3), (1999), 61-88.
9. J.-C. Faugère, A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5), *In Proceedings of the 2002 international symposium on symbolic and algebraic computation, ISSAC 2002, Lille, France, July 07–10, 2002. New York, ACM Press*, (2002), 75-83.
10. R. Gebauer, H. Möller, On an Installation of Buchberger's Algorithm, *J. Symb. Comput.*, **6**(2-3), (1988), 275-286.
11. A. Hashemi, M. Dehghani Darmian, M. Barkhordar, Gröbner Systems Conversion, *Math. Comput. Sci.*, **11**(1), (2017), 61-77.

The Maple codes of the algorithms are available at `http://amirhashemi.iut.ac.ir/softwares` under the names `PF4.mpl`, `PLA-PFGLM.mpl` and `Montes.mpl`. The first file contains a MAPLE implementation of our algorithm for computing Gröbner systems. The second file contains a MAPLE implementation of the Kapur et al. algorithm (PGBMAIN algorithm) and the last one is a MAPLE implementation of the improvement of the Montes DISPGB algorithm.

[DOI: 10.61186/ijmsi.19.1.117]

12. A. Hashemi, M. Dehghani Darmian, B. M.-Alizadeh, Applying Buchberger's Criteria on Montes's DisPGB Algorithm, *Bull. Iran. Math. Soc.*, **38**(3), (2012), 715-724.
13. A. Hashemi, B. M.-Alizadeh, M. Dehghani Darmian, Minimal Polynomial Systems for Parametric Matrices, *Linear Multilinear Algebra*, **61**(2), (2013), 265-272.
14. M. Kalkbrener, On the Complexity of Gröbner Bases Conversion, *J. Symb. Comput.*, **28**(1-2), (1999), 265-273.
15. D. Kapur, An Approach for Solving Systems of Parametric Polynomial Equations, *In Saraswat, Vijay, Van Hentenryck, Pascal (Eds.), Principles and Practice of Constraint Programming. MIT Press*, (1995), 217-224.
16. D. Kapur, Y. Sun, D. Wang, A New Algorithm for Computing Comprehensive Gröbner Systems, *In Proceedings of the 35th international symposium on symbolic and algebraic computation, ISSAC 2010, Munich, Germany, July 25–28, 2010. New York, NY: Association for Computing Machinery (ACM)*, (2010), 29-36.
17. D. Kapur, Y. Sun, D. Wang, An Efficient Algorithm for Computing a Comprehensive Gröbner System of a Parametric Polynomial System, *J. Symb. Comput.*, **49**, (2013), 27-44.
18. D. Lazard, Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations, *Computer algebra, EUROCAL '83, Proc. Conf., London 1983, Lect. Notes Comput. Sci.*, **162**, (1983), 146-156.
19. M. Manubens, A. Montes, Improving the DisPGB Algorithm Using the Discriminant Ideal, *J. Symb. Comput.*, **41**(11), (2006), 1245-1263.
20. M. Manubens, A. Montes, Minimal Canonical Comprehensive Gröbner Systems, *J. Symb. Comput.*, **44**(5), (2009), 463-478.
21. A. Montes, A New Algorithm for Discussing Gröbner Bases with Parameters, *J. Symb. Comput.*, **33**(2), (2002), 183-208.
22. A. Montes, *The Gröbner Cover*, **27**, Cham: Springer, 2018.
23. A. Montes, J. Castro, Solving the Load Flow Problem Using the Gröbner Basis, *SIGSAM Bull.*, **29**(1), (1995), 1-13.
24. A. Montes, M. Wimber, Gröbner Bases for Polynomial Systems with Parameters, *J. Symb. Comput.*, **45**(12), (2010), 1391-1425.
25. K. Nabeshima, A Speed-up of the Algorithm for Computing Comprehensive Gröbner Systems, *In Proceedings of the 2007 international symposium on symbolic and algebraic computation, ISSAC 2007, Waterloo, ON, Canada, July 29–August 1, 2007. New York, NY: Association for Computing Machinery (ACM)*, (2007), 299-306.
26. W. Y. Sit, M. Wimber, An Algorithm for Solving Parametric Linear Systems, *J. Symb. Comput.*, **13**(4), (1992), 353-394.
27. A. Suzuki, Y. Sato, A Simple Algorithm to Compute Comprehensive Gröbner Bases Using Gröbner Bases, *In Proceedings of the 2006 international symposium on symbolic and algebraic computation, ISSAC 06, Genova, Italy, July 9–12, 2006. New York, NY: Association for Computing Machinery (ACM)*, (2006), 326-331.
28. V. Weispfenning, Comprehensive Gröbner Bases, *J. Symb. Comput.*, **14**(1), (1992), 1-29.
29. V. Weispfenning, Canonical Comprehensive Gröbner Bases, *J. Symb. Comput.*, **36**(3-4), (2003), 669-683.
30. M. Wimber, Gröbner Bases for Families of Affine or Projective Schemes, *J. Symb. Comput.*, **42**(8), (2007), 803-834.

[DOI: 10.61186/ijmsi.19.1.117]