Solving Polynomial Systems over Finite Fields: Algorithms, Implementation and Applications

Chenqi Mou

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique et Mathématique

Ecole Doctorale Informatique, Télécommunications et Électronique (Paris)

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Présentée et soutenue par

Chenqi Mou

Solving Polynomial Systems over Finite Fields : Algorithms, Implementations and Applications

Thèse dirigée par Jean-Charles Faugère et Dongming Wang
préparée au LIP6, UPMC et SMSS, Beihang University

Soutenue le 29 mai 2013 après avis des rapporteurs :

Masayuki Noro	Professeur, Kobe University
Éric Schost	Professeur associé, University of Western Ontario

devant le jury composé de :

Jean-Charles Faugère	Directeur de Recherche, INRIA Paris-Rocquencourt
Shangzhi Li	Professeur, Beihang University
Dongdai Lin	Professeur, Institute of Information Engineearing, CAS
Mohab Safey El Din	Professeur, Université Pierre et Marie Curie
Éric Schost	Professeur associé, University of Western Ontario
Dingkang Wang	Professeur, Academy of Mathematics and Systems Science, CAS
Dongming Wang	Directeur de Recherche, CNRS
Meng Zhou	Professeur, Beihang University

To My Family.

Abstract

Polynomial systems can be used to formulate a large variety of non-linear problems. Polynomial systems over finite fields are of particular interest because of their applications in Cryptography, Coding Theory, and other areas of information science and technologies. Solving polynomial systems is a central topic in Computer Algebra and has stimulated the development in this area. In this thesis, theories, algorithms, implementations and applications of solving polynomial systems over finite fields are studied, leading to several major contributions as follows.

(1) We propose efficient algorithms for changing the orderings of Gröbner bases of zero-dimensional ideals to LEX by making use of the sparsity of multiplication matrices and study their complexities. For ideals in shape position, a probabilistic algorithm is proposed to recover the LEX Gröbner basis by computing the minimal polynomial of a linearly recurring sequence generated by one multiplication matrix and by solving a structured linear system. Deterministic and incremental variants of this algorithm are also presented for related but different purposes. For general ideals, we generalize the above principle by defining linearly recurring relations with all the multiplication matrices, and compute the LEX Gröbner basis as the generating polynomial set w.r.t. LEX by the BMS algorithm from Coding Theory. Computational complexities of all these algorithms and the sparsity of one multiplication matrix for generic polynomial systems are analyzed, and the effectiveness and efficiency of the algorithms are verified by our implementations.

(2) We present algorithms for decomposing polynomial sets over finite fields into simple sets. For zero-dimensional polynomial sets, an algorithm is designed on the basis of generalization of squarefree decomposition of univariate polynomials over finite fields and a specialized technique for computing the pth root of any element in products of field extensions determined by simple sets; In the positive-dimensional case, we reduce simple decomposition to computation of radical ideals of positive characteristics and positive dimensions, and then make use of existing algorithms for computing such radicals to propose an effective algorithm. We have implemented both algorithms and preliminary experimental results are also provided.

(3) We study the problems of squarefree decomposition and factorization for polynomials over unmixed products of field extensions represented by simple sets and propose algorithmic solutions for them using multiple derivations and transformations of ideals to those in shape position. Examples and experimental results are provided to illustrate the presented algorithms and to show their performances.

(4) We apply methods for solving polynomial systems over finite fields to detect the steady states and their numbers of finite biological models and to find the minimal polynomial in the interpolation problem in Sudan's list decoding algorithm in Coding Theory. In the first application we mainly adopt a traversal method based on Gröbner bases and a method based on specialized algorithm for triangular decomposition over $\mathbb{F}_2$; In the second application we reduce the original problem to change of ordering of Gröbner bases, for which the algorithm for change of ordering proposed in this thesis is potentially applicable.

Key Words: Polynomial system, finite field, Gröbner basis, triangular set, change of ordering, simple set, unmixed product of field extensions

Acknowledgements

First I would like to express my deepest gratitude to both of my supervisors Dongming Wang and Jean-Charles Faugère for introducing me to such a research subject of interest, beauty and challenges. Their priceless guidance and help have accompanied me throughout my entire PhD study, and their wisdom and patience have enlightened this important period of my life. What I have learnt from you is the treasure that will benefit my whole life!

It is an honor to have Masayuki Noro and Éric Schost to review my thesis. Their time and careful reading are highly appreciated, and their detailed and professional comments on the thesis have led to great improvement on it. And I also want to express my warm thanks to Shangzhi Li, Dongdai Lin, Mohab Safey El Din, Éric Schost, Dingkang Wang, Meng Zhou for attending the thesis defense as jury members.

During my PhD study, LIP6, UPMC, France and SMSS–LMIB, Beihang University, China have provided me with excellent research environment. I am grateful to the researchers Daniel Lazard, Mohab Safey El Din, Ludovic Perret, Guénaël Renault, Zhiming Zheng, Tiegang Liu, Meng Zhou, Zhikun She, Xiaoyuan Yang, and PhD students and postdocs Luk Bettale, Christopher Goyet, Mourad Gouicem, Aurélien Greuet, Louise Huot, Pierre-Jean Spaenlehauer, Mate Soos, Xiaoyu Chen, Yanli Huang, Ying Huang, Meng Jin, Tielin Liang, Ye Liang, Tian Luo, Ying Shao, Jing Yang, Ting Zhao from these two institutions for their help. Especially, I would like to thank Xiaoliang Li and Wei Niu for their helpful discussions in our cooperations.

I owe my thanks to Chinese Scholarship Council for the financial support during my study in France and to Service de l'Education Ambassade de la Réublique Populaire de Chine en Réublique Françise for providing a warm and friendly environment to Chinese students in France.

I also would like to thank my friends Kai Shi, Zhiyong Sun, Yunlong Tan, Wei Wang, Xiaoliang Fan, Sheng Gao, Xin Lin, Yan Ma, Tianze Wang, Xiaomin Wang, Yan Zhang both in China and France for accompanying and supporting me during the thesis.

Last but not least, I feel grateful to my wife Lin Zhang and my family. The thesis is dedicated to you for your endless love and support.

Table of Contents

Introduction

Polynomial system solving: an overview

Polynomial systems are fundamental objects that appear in numerous areas of science and engineering such as Commutative Algebra [35], Algebraic Geometry [36], Geometric Reasoning [164], Algebraic Cryptanalysis [54], and Signal Processing [102]. In this thesis we are interested in *polynomial system solving*, a central topic which has stimulated and will definitely continue stimulating the development of the subject *Computer Algebra* [14, 66, 85, 109, 153, 163].

To solve polynomial systems, one needs not only to study the underlying theory of polynomial ideals and varieties in Commutative Algebra and Algebraic Geometry, but also to design and implement constructive methods with proven correctness and termination. In particular, as a remarkable characteristic of methods in Computer Algebra, we use *symbolic* methods to pursue the *exact* solutions of polynomial systems with no error.

The problem of solving polynomial systems can be formulated in the following way. Let $\mathbb{K}$ be a field and $\mathcal{F}$ be any multivariate polynomial set over $\mathbb{K}$ with respect to (short as w.r.t. hereafter) the variables $x_1, \ldots, x_n$, and $\tilde{\mathbb{K}}$ be some field extension of $\mathbb{K}$. Then $\overline{\boldsymbol{x}} \in \tilde{\mathbb{K}}^n$ is said to be a *solution* of $\mathcal{F} = 0$ in $\tilde{\mathbb{K}}$ (may be omitted when $\tilde{\mathbb{K}}$ is $\overline{\mathbb{K}}$, the algebraic closure of $\mathbb{K}$) if $\mathcal{F}(\overline{\boldsymbol{x}}) = 0$, and solving the polynomial system $\mathcal{F} = 0$ means to *compute* all the solutions of $\mathcal{F} = 0$ in $\tilde{\mathbb{K}}$, and *represent* the solutions in an appropriate way. It is remarkable that when solving $\mathcal{F} = 0$, usually we are only

interested in its *solutions* without their *multiplicities*, and in this case indeed it is sufficient to study the radical ideal $\sqrt{\langle \mathcal{F} \rangle}$.

The polynomial system $\mathcal{F} = 0$ is said to be *zero-dimensional* if it has finitely many solutions (in this case we also call $\mathcal{F}$ zero-dimensional). For zero-dimensional polynomial sets, representation of their solutions means the enumeration of all of them. In particular, in the case $\mathbb{K}$ is a finite field, solving $\mathcal{F} = 0$ *in* $\mathbb{K}$ means to enumerate its solutions in $\mathbb{K}$, which are obviously finite.

Triangular sets are widely considered to be good representations for the solutions of polynomial systems. A triangular set $\mathcal{T}$ in $\mathbb{K}[x_1, \ldots, x_n]$ $(x_1 < \cdots < x_n)$ is an ordered set of polynomials in the form $[T_1(x_1, \ldots, x_{p_1}), \ldots, T_r(x_1, \ldots, x_{p_r})]$ with $1 \leq p_1 < \cdots < p_r \leq n$. In the case of $r = n$, $\mathcal{T}$ is zero-dimensional and all its solutions can be computed by successively solving univariate polynomials after substitutions of variables by preceding solutions; otherwise the triangular form of $\mathcal{T}$ also makes it easy to study the properties of the solutions of $\mathcal{T}$, for example the *parameters*, analogy to the *free variables* in linear systems, can be read directly.

The mainstream methods for solving polynomial systems are first summarized in Figure 1. Besides semi-numeric methods like the homotopy continuation method [152], there are two major methods in Computer Algebra based on respectively Gröbner bases [14, 111] and triangular sets [159, 5], through which polynomial sets are reduced to new ones with certain triangular structures and the same solutions. By elimination of variables the problem for solving polynomial systems is turned into that of solving univariate polynomial equations in a symbolic way [127, 86].

By the method of triangular sets the polynomial set $\mathcal{F}$ of interest is decomposed into finitely many triangular sets such that the zeros of $\mathcal{F}$ is precisely the union of the zeros of all these triangular sets (this process is called *triangular decomposition*) [159, 5]. Hence, after the triangular decomposition one can acquire the solutions of $\mathcal{F} = 0$ by computing the zeros of each associated triangular set, which is relatively easy because of its triangular structure [90, 77, 112]. Indeed in the process of triangular decomposition the multiplicities of the solutions are disregarded.

Gröbner bases w.r.t. the *lexicographical* ordering (short as LEX) have the strongest algebraic structure convenient for polynomial system solving, so to solve the polynomial system $\mathcal{F} = 0$ by the method of Gröbner bases, we need to compute the LEX

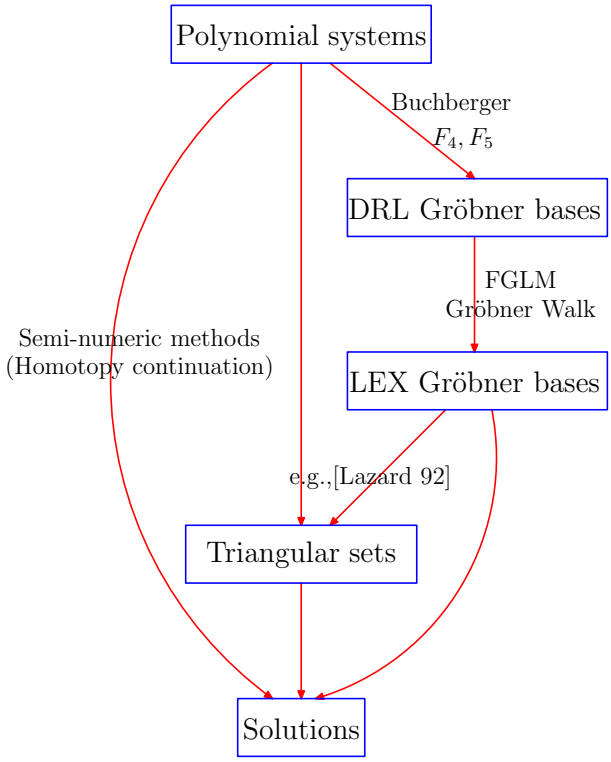

Figure 1: Solving polynomial systems: an overview

Gröbner basis of $\langle \mathcal{F} \rangle$. Compared with direct computation of the LEX Gröbner basis with algorithms like F_4 or F_5 [49, 50], it is usually more efficient to first compute a Gröbner basis of $\langle \mathcal{F} \rangle$ w.r.t. the *degree reversed lexicographical* ordering (short as DRL) [21, 49, 50], and convert it to a Gröbner basis w.r.t. LEX with the FGLM algorithm [53, 55] or the *Gröbner walk* [29]. Since the *Shape Lemma* holds for the ideals such that the first coordinates of any two zeros do not coincide [13] (in particular, for ideals generated by *generic* polynomial systems this property holds), namely the LEX Gröbner bases of these ideals are of the form $G_1(x_1), x_2 + G_2(x_1), \ldots, x_n + G_n(x_1)$ (such ideals are said to be in *shape position*), it can be concluded that in most cases the computed Gröbner bases w.r.t. LEX are triangular sets. Otherwise one may choose to turn the LEX Gröbner basis to triangular sets to represent the solutions [91]. Though more steps may be needed to reach the solutions of the polynomial systems, the method by Gröbner bases has evident advantages, e.g., the efficiency.

Gröbner bases

The concept of Gröbner bases was first introduced by B. Buchberger [19, 21]. Then an extensive amount of study has been performed on the Gröbner basis concerning its theory, algorithms, and applications. Now the Gröbner basis theory has become a relatively mature subject, with many academic monographs and textbooks on it (e.g., [14, 35, 153]).

Given a term ordering $<$, we denote the leading term of a polynomial $F \in \mathbb{K}[x_1, \ldots, x_n]$ w.r.t. $<$ by $\mathrm{lt}(F)$ if no ambiguity occurs. Then for any ideal $\mathfrak{a} \subset \mathbb{K}[x_1, \ldots, x_n]$, its *Gröbner basis* w.r.t. $<$ is a set of polynomials $G_1, \ldots, G_r \in \mathfrak{a}$ such that $\langle \mathrm{lt}(F) : F \in \mathfrak{a} \rangle = \langle \mathrm{lt}(G_1), \ldots, \mathrm{lt}(G_r) \rangle$. Many theoretical and practical problems related to polynomial ideals can be solved with Gröbner bases in a computational way, for example the membership test problem (see Section 1.2) [35].

One remarkable characteristic of Gröbner bases is their computability. The Buchberger algorithm, S-pair criterion and corresponding optimization criteria form the framework of the initial study on the Gröbner basis theory [20, 21]. Then J.-C. Faugère proposed the algorithms F_4 which makes use of the relationship between the Buchberger algorithm and linear algebra [49] and F_5 in which useless S-pair selection is avoided [50]. These two algorithms greatly improve the Buchberger algorithm to achieve huge efficiency gains. Following researches include, for example, F_5 algorithm variants [46] and the study on signature related algorithms for Gröbner basis computation [147].

In the worst case the complexity to compute Gröbner bases is proven to be double exponential in the number of variables [108], but for *generic* or *structured* systems, the complexities are much lower. A homogeneous nonzero polynomial sequence $\mathcal{F} = [F_1, \ldots, F_r] \subset \mathbb{K}[x_1, \ldots, x_n]$ is said to be *regular* if for each $i = 2, \ldots, r$, F_i is not a zero-divisor in the quotient ring $\mathbb{K}[x_1, \ldots, x_n]/\langle F_1, \ldots, F_{i-1} \rangle$, and roughly speaking *semi-regular* sequences extend regular ones by focusing on the case when there are more polynomials than the variables in $\mathbb{K}[x_1, \ldots, x_n]$. It has been shown that for any semi-regular sequence in $\mathbb{K}[x_1, \ldots, x_n]$, the DRL Gröbner basis of the ideal generated by it can be computed by the F_5 algorithm within $O\left(\binom{n+d_{reg}}{n}^{\omega}\right)$ arithmetic operations in $\mathbb{K}$, where d_{reg}, called *degree of regularity* of the system, is the highest degree reached in the computation of Gröbner basis in the F_4 algorithm, and $\omega < 2.39$ is the exponent

in the complexity of efficient matrix multiplication [11]. Complexities of Gröbner basis computations for structured systems like bilinear, determinantal, and Boolean ones are studied in [143].

One important step for efficiently solving polynomial systems is to transform Gröbner bases w.r.t. DRL to those w.r.t. LEX. There have been effective methods for changing the orderings of Gröbner bases, for example the FGLM algorithm for the zero-dimensional case [53] or the Gröbner walk for the general case [29]. Related works include a dedicated algorithm for bivariate zero-dimensional ideals based on the LLL algorithm [12] and an algorithm for computing *Gröbner fans* which are useful in the Gröbner Walk [63]. However, this crucial step of changing the orderings has become the bottleneck for solving large polynomial systems from many applications, for example when the number of solutions is greater than 1000.

Triangular sets

J.F. Ritt first introduced the notion of characteristic sets, which are a special kind of triangular sets [134]. Then W.-T. Wu extended Ritt's work and greatly developed the characteristic set method in theory, algorithms, efficiency and applications [164, 165]. The general theory of triangular sets can be found in [5, 74, 159].

For any polynomial $F \in \mathbb{K}[x_1, \ldots, x_n]$, it can be written in the form $F = Ix_k^d + R$, where x_k is the greatest variable appearing in F and the degree of R w.r.t. x_k is smaller than d. Then the polynomial I is called the *initial* of F, denoted by $\mathrm{ini}(F)$. For any triangular set $\mathcal{T} \subset \mathbb{K}[x_1, \ldots, x_n]$, the ideal $\langle \mathcal{T} \rangle : H^\infty$, where $H = \prod_{T \in \mathcal{T}} \mathrm{ini}(T)$, is called the *saturated ideal* of $\mathcal{T}$, denoted by $\mathrm{sat}(\mathcal{T})$. These two concepts are useful in the theory of triangular sets, for example we study frequently the quotient ring $\mathbb{K}[x_1, \ldots, x_n]/\mathrm{sat}(\mathcal{T})$.

Besides characteristic sets [27, 156, 23, 64], one kind of remarkable triangular sets are the *regular sets*, also known as *regular chains* [77] and *proper ascending chains* [167]. Regular sets have a good property that the initial of each polynomial in them does not vanish at the zeros determined by the preceding truncated regular sets. In particular, a generalized GCD computation called *pseduo-gcd*, or GCD modulo regular sets, is introduced and studied [113, 78, 74, 113, 95], which will be extensively used in the thesis. It worths mentioning that this pseudo-gcd makes use of the *D5 principle*

(or called *dynamic evaluation*) to introduce dynamic splitting [42, 45, 16, 121].

Each polynomial in an *irreducible* triangular set is irreducible over the preceding field extension defined by the truncated irreducible triangular set [155]. Irreducible triangular sets are the standard representations for finitely generated field extensions. Clearly, to design algorithms for irreducible decompositions, one needs to realize factorization over finitely generated field extensions [39, 160, 75, 146].

Among various kinds of triangular sets, *simple sets* are characterized by the property that each of their polynomials is conditionally squarefree. The notion of simple sets originates from [149, 157] and is the same as the so-called *squarefree regular chains* [74]. Algorithms for simple decomposition over fields of 0 and positive characteristics have been presented in [8, 98, 99]. Simple sets have also been used to count the solutions of polynomial systems [133].

All the above triangular sets have been extended to corresponding triangular systems by D. Wang [158, 157], where polynomial inequalities are introduced. In particular, a tight connection between simple systems and dynamic constructible closures has been revealed [43].

Efficient implementations of algorithms for triangular decompositions are also an active research focus [156, 6, 93, 96]. Like the Gröbner bases, changing the variable orderings of triangular sets has also been studied [130].

Applications of polynomial system solving

The theory and methods of polynomial system solving have been widely used in various areas. In the following we provide several applications as motivations of our study on polynomial system solving.

Geometric theorem proving: In the 1970s W.-T. Wu proposed an automatic method to prove elementary geometric theorems. His method essentially reduces the proof of geometric theorems to determining the relationship of the zeros of the algebraic equations defined by the hypotheses and conclusions, and the reduced problem is further solved with the tool of triangular sets [164, 165]. Then hundreds of non-trivial theorems are proved with implementations of Wu's method [26, 154]. The development of this method in geometric theorem proving are summarized in [25, 166]. It is shown that Gröbner bases can also be used to reason about geometric problems [80].

Cryptography: It has been proved that solving multivariate polynomial equation sets over finite fields is NP-hard [65]. Based on this difficulty, a category of cryptosystems has been designed: *multivariate public-key cryptosystems* [44], with well-known schemes like Matsumoto-Imai, Hidden Field Equations, Oil–Vinegar, and Tame Transformation Method [107, 132, 110]. The central mappings of multivariate public-key schemes are usually structured polynomial sets which are easy to evaluate but hard to solve, and thus the study on specialized methods to solve structured polynomial systems are stimulated [131, 33, 57].

Another application of polynomial system solving in cryptography is the *algebraic cryptanalysis* on existing cryptosystems, which mainly focuses on breaking a cryptosystem or analyzing its security by algebraic methods [9, 34, 103]. One possible method is solving the polynomial system over a finite field reduced and simplified from a cryptosystem. Methods based on Gröbner bases, triangular sets, and SAT solvers for polynomial system solving have been successfully applied to break some proposed cryptosystems (e.g., [54, 73, 142]).

It should be remarked that in the application to cryptography, polynomial system solving over finite fields is particularly important because the general setting is the finite field.

Coding Theory: Gröbner bases have been used to decode specific codes in Coding Theory. For example, the papers [24, 104] present algorithms for decoding cyclic codes up to the true minimal distance with Gröbner bases, and [60, 61, 92] show how to use Gröbner bases to decode Reed–Solomon codes.

Algebraic-geometric codes, initialized by the geometric Goppa code [69], are a kind of error-correcting codes derived from algebraic curves [72]. The decoding of algebraic-geometric codes with Gröbner bases is studied in [71, 137]. In particular, the Berlekamp–Massey–Sakata (BMS for short) algorithm is a useful tool for decoding such codes, and it shows a very strong connection with Gröbner bases [138, 139].

Reviews on the application of Gröbner bases in Coding Theory may be found in [140, 36].

Algebraic Biology: The successful application of algebraic methods, including those from Computer Algebra, in biology has been influential [126, 88], leading to the advent of a new interdisciplinary field called Algebraic Biology. A series of conference

"Algebraic and Numeric Biology" has arisen in response to this need. One instance of the use of polynomial system solving in Algebraic Biology is detecting stable points of continuous and discrete biological models [120, 97].

Of course there are a lot of other applications of polynomial system solving, for example in Control Theory [129], Robotics [35, 100], Integer Programming [31, 150], etc. Besides the theoretical significance of the study on how to solve polynomial systems, these practical applications also motivate this thesis.

Problem statement

We mainly focus on the following problems related to polynomial system solving over finite fields (See Figure 1 for the corresponding relationship).

(**Sparse FGLM algorithms**) From the computational point of view, with recent progress on Gröbner basis computation [49, 50], changing the orderings of Gröbner bases of zero-dimensional ideals has become the bottleneck of the whole process for solving polynomial systems with Gröbner bases. Hence, it is of crucial significance to design efficient algorithms to perform the change of ordering.

For any zero-dimensional ideal $\mathfrak{a}$ in $\mathbb{K}[x_1, \ldots, x_n]$ of degree D, the multiplication matrix w.r.t. x_i in the FGLM algorithm is a $D \times D$ matrix to represent the linear map $f : \mathbb{K}[x_1, \ldots, x_n]/\mathfrak{a} \to \mathbb{K}[x_1, \ldots, x_n]/\mathfrak{a}$ defined by $f(G) = x_i G$. In the FGLM algorithm, multiplication matrices link the change of ordering of Gröbner bases to linear algebra operations. In particular, they usually possess a sparse structure, even when the defining polynomial sets are dense.

Problems: (1) *Make use of the inherent sparsity of multiplication matrices to design efficient algorithms for changing the orderings of Gröbner bases of zero-dimensional ideals.* (2) *Analyze the complexities of the proposed algorithms.* (3) *Evaluate the sparsity of multiplication matrices for generic polynomial systems.*

(**Simple decomposition over finite fields**) A simple set is a special regular set in which every polynomial is conditionally squarefree w.r.t. its leading variable. One of its remarkable properties is that the number of its zeros (all of multiplicity 1) in the algebraic closure of the ground field $\mathbb{K}$ can be easily counted by looking at the leading degrees of its polynomials. Compared with an irreducible triangular set which defines a finite extension of $\mathbb{K}$, a simple set may represent a product of such

field extensions. The representation of field extensions is clearly made more compact by using one simple set than using several irreducible triangular sets.

Algorithms for decomposing polynomial sets into simple sets have been proposed by Wang [157] and by Bächler and others [8] for the case in which $\mathbb{K}$ is of characteristic 0. However, to our best knowledge, there is no algorithm available for *simple decomposition* over finite fields, namely decomposing polynomial sets over finite fields into simple sets.

Problem: *Design effective algorithms for simple decomposition over finite fields, and implement them.*

(**Unmixed products of field extensions**) By product of field extensions we mean a direct product of k finitely generated field extensions of a base field $\mathbb{K}$. It is a nontrivial generalization of the concept of field extension when $k > 1$. Polynomials over finitely generated field extensions are fundamental objects in commutative algebra, and so are polynomials over products of such field extensions. Efficient operations with polynomials over products of field extensions allow one to deal with computational tasks such as decomposition, combination, and relationship testing for polynomial ideals and algebraic varieties effectively [37, 41, 98, 155, 157].

For any simple set $\mathcal{S} \subset \mathbb{K}[x_1, \ldots, x_n]$, the quotient ring of $\mathbb{K}[x_1, \ldots, x_n]/\operatorname{sat}(\mathcal{S})$ is isomorphic to a product of field extensions of $\mathbb{K}$ [98]. This observation motivates the following problems.

Problems: (1) *Identify which category of products of field extensions corresponds to simple sets.* (2) *Generalize squarefree decomposition and factorization on polynomials over fields to those over such products of field extensions by using simple sets, and design corresponding algorithms for them.*

(**Applications in Biology and Coding Theory**) Finite dynamical systems are usually used to model biological phenomena, and they are simply structured and have the advantage of input of a small size. Since it is difficult to find the analytical solution of a finite biological model (if such solutions exist at all), the detection of their steady states and the numbers becomes important to study the qualitative behaviors of their solutions, and thus their dynamical characteristics.

Problem: *Compute the steady states and their numbers for biological systems modeled as first-order finite difference equations with algebraic tools like Gröbner bases*

and triangular sets.

In Coding Theory, the list decoding is a useful method for decoding error-correcting codes of large error rates. One of the two steps in the list decoding algorithms proposed in [70, 84] is to find the minimal bivariate polynomial w.r.t. a given term ordering which passes through given two-dimensional points with at least the given multiplicities.

Problem: *Find this minimal polynomial in the list decoding problem with efficient algorithms for change of ordering of Gröbner bases.*

Contributions of the thesis

In this thesis we aim at answering those problems raised above. To be precise, we study several important theoretical and computational aspects for solving polynomial systems over finite fields and propose original algorithms for fast change of ordering of Gröbner bases, for simple decomposition over finite fields, and for squarefree decomposition and factorization over unmixed products of field extensions. Methods for polynomial system solving over finite fields are also applied to solve practical problems arising from Biology and Coding Theory.

Sparse FGLM algorithms

Let $\mathfrak{a}$ be any zero-dimensional ideal of degree D in $\mathbb{K}[x_1, \ldots, x_n]$. We study the problem of changing the orderings of Gröbner bases from a given term ordering to LEX by using the sparsity of multiplication matrices.

- (**Ideals in shape position: probabilistic algorithm**) *Suppose $\mathfrak{a}$ is in shape position. A probabilistic algorithm is proposed to change the ordering of its Gröbner basis to LEX with the complexity $O(D(N_1 + n\log(D)^2))$, where N_1 is the number of nonzero entries in the multiplication matrix T_1.*

  It is easy to see that the univariate polynomial in the LEX Gröbner basis of $\mathfrak{a}$ may be computed by applying Berlekamp–Massey algorithm [162] to the constructed linearly recurring sequence $[\langle \boldsymbol{r}, T_1^i \boldsymbol{e}\rangle : i = 0, \ldots, 2D-1]$, where $\boldsymbol{r}$ is a random vector, $\boldsymbol{e} = (1, 0, \ldots)^t$ is the vector representing $\mathbf{1}$ in $\mathbb{K}[x_1, \ldots, x_n]/\mathfrak{a}$, and $\langle \cdot, \cdot \rangle$ takes the inner product. This proposed method is characterized by

the way we recover the other polynomials in the Gröbner basis: constructing and solving structured linear systems with Hankel coefficient matrices.

- (**Ideals in shape position: deterministic algorithm**) *Suppose* $\mathfrak{a}$ *is in shape position. A deterministic algorithm is proposed to return the LEX Gröbner basis of* $\sqrt{\mathfrak{a}}$ *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* $|\mathbb{K}| = q$*.*

  As discussed previously, in the setting of polynomial system solving, the radical ideal $\sqrt{\mathfrak{a}}$ is usually of interest. This proposed method uses the deterministic Wiedemann algorithm to find the univariate polynomial in the LEX Gröbner basis. Then by the Chinese Remainder Theorem we adapt and extend the probabilistic algorithm to recover the Gröbner basis of $\sqrt{\mathfrak{a}}$, instead of $\mathfrak{a}$.

  In order to design an algorithm to compute the univariate polynomial in the LEX Gröbner basis which has a complexity sensitive to the degree of the output polynomial, we also briefly discuss how to apply an incremental variant of the Wiedemann algorithm to compute the univariate polynomial, and this variant is efficient when the degree of the output polynomial is small.

- (**General ideals**) *A probabilistic algorithm is proposed to change the ordering of Gröbner bases of general ideals with the complexity* $O(nD(N+\hat{N}\bar{N}D))$*, where* N *is the maximal number of nonzero entries in matrices* $T_1, \ldots, T_n$*, while* $\hat{N}$ *and* $\bar{N}$ *are respectively the number of polynomials and the maximal term number of all polynomials in the resulting Gröbner basis.*

  We generalize the linearly recurring sequence to an n-dimensional mapping $\mathrm{E} : (s_1, \ldots, s_n) \longmapsto \langle \boldsymbol{r}, T_1^{s_1} \cdots T_n^{s_n} \boldsymbol{e} \rangle$. The minimal set of generating polynomials w.r.t. LEX for the linearly recurring relation determined by E is essentially the LEX Gröbner basis of the ideal defined by E. We can compute this minimal generating set with the BMS algorithm from Coding Theory [138, 139], and this set coincides with our target LEX Gröbner basis with probabilities.

  Combining all these methods above, we present a deterministic algorithm for change of ordering of Gröbner bases of zero-dimensional ideals, which is able to choose automatically which sub-algorithm to use according to the input.

- (**Sparsity of multiplication matrices and refined algorithm complexity**) Assuming the correctness of the Moreno-Socías conjecture [114], we are able to prove the structure which the terms in $\mathrm{lt}(\langle \mathcal{P} \rangle)$ satisfy for any *generic* polynomial system $\mathcal{P}$ w.r.t. DRL: for a term $\boldsymbol{u} \in \mathrm{lt}(\langle \mathcal{P} \rangle)$, any term $\boldsymbol{v}$ such that $\deg(\boldsymbol{u}) = \deg(\boldsymbol{v})$ and $\boldsymbol{v} >_{\mathsf{DRL}} \boldsymbol{u}$ is also in $\mathrm{lt}(\langle \mathcal{P} \rangle)$ (see Figure 2 below for the bivariate case).

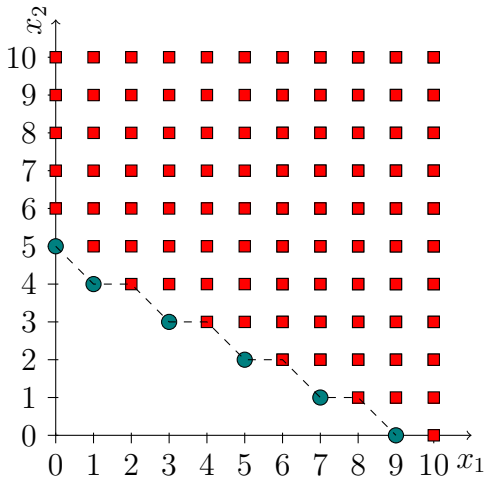

Figure 2: Minimal generators of $\mathrm{lt}(\langle \mathcal{P} \rangle)$ (●) and terms in $\mathrm{lt}(\langle \mathcal{P} \rangle)$ (■)

Then for generic polynomial systems in n variables of degree d, we have the following results on the sparsity of the multiplication matrix T_1 and the refined complexity of the probabilistic algorithm for ideals in shape position.

(a) *If the Moreno-Socías conjecture holds, then the number of dense columns in the multiplication matrix T_1 is equal to the greatest coefficient in the expansion of $(1 + z + \cdots + z^{(d-1)})^n$.*

(b) *Let n be fixed. If the Moreno-Socías conjecture holds, then as d tends to $+\infty$, the percentage of nonzero entries in T_1 is $\sim \sqrt{\frac{6}{n\pi}}/d$, and for a generic polynomial system of degree d, the complexity of the probabilistic algorithm for ideals in shape position is $O(\sqrt{\frac{6}{n\pi}} D^{2+\frac{n-1}{n}})$.*

- (**Experimental results**) The efficiency of the proposed methods is verified by experiments with our implementations. The timings for selected benchmarks are presented in Tables 1 and 2 below. It can be concluded that for ideals

in shape position, the current implementation outperforms those FGLM ones in Magma and Singular. In addition, for Katsura and randomly generated systems, compared to the actual density of T_1, the theoretical bound works and the asymptotic estimation is also close.

		Matrix Density			FGb		Magma		Singular		Speedup	
Name	D	Actual	Theoretical	Asymp	F_5(C)	Sparse FGLM	F_4	FGLM	Buchberger	FGLM	Magma	Singular
Random 12,d=2	2^{12}	21.26%	22.56%	19.95%	19.9s	10.7s	134.9	1335.8	4867.4	2581.1	124.8	241.2
Random 13,d=2	2^{13}	19.98%	20.95%	19.16%	118.0s	80.8s	949.6	10757.4	36727.0	19820.2	133.1	245.3
Random 4,d=8	4096	7.54%	8.40%	8.64%	2.5s	4.2s	3.5s	556.7s			132.5	
Random 4,d=9	6561	6.66%	7.45%	7.68%	6.6s	13.5s	9.3s	1800.9s			133.4	
Random 3,d=17	4913	3.92%	4.42%	4.69%	1.5s	4.4s	1.95s	585.3s			133.0	
Random 3,d=18	5832	3.70%	4.17%	4.43%	2.3s	6.5s	2.9s	1142.6s			175.8	
Katsura13	2^{13}	19.86%	20.95%	19.16%	177.2s	80.9s	1037.2s	10895.4s			134.7	
Katsura14	2^{14}	19.64%	20.95%	18.47%	1285.1s	553.4 s	9599.0s	87131.9s			157.4	
Eco 12 ‡	1024	29.69%	NA	NA	2.8s	0.35s	29.6s	456.5s			1304	
Eco 13 ‡	2048	27.52%	NA	NA	15.3s	2.0s	262.7s	5692.7s			2846	
Crit D=4,p=2,n=6	6480	14.11%	NA	NA	16.3s	27.3s	63.2s	3196.7s			117.1	
Crit D=5,p=2,n=5	6400	11.00%	NA	NA	10.2s	19.9s	27.9s	2335.1s			117.3	
MinR(9,6,3)	980	26.82%	NA	NA	0.7s	0.3s	6.3s	22.7s	137.5s	38.1s	75.7	127.0
MinR(9,7,4)	4116	22.95%	NA	NA	17.5s	10.8s	208.1s	1360.4s	4985.8s	2490.3s	126.0	230.6
DLP EC n=4	4096	7.54%	NA	NA	2.4s	4.25s	7.4s	475.8s	72.4s	1823.6s	112.0	429.1
DLP Edwards n=4	512	7.68%	NA	NA	0.03s	0.02s	0.16s	21.4s			1066	
Cyclic 10 †	34940	1.00%	NA	NA		1107.8s	>16 hrs and >16 Gig					

Table 1: Selected timings for ideals in shape position (from DRL to LEX)

Name	n	D	Mat Density	Poly Density	#Passes	#Mat.	#Red.	#Multi.
Cyclic5	5	70	12.02%	21.13%	499	1347	421	$nD^{2.702}$
Cyclic6	6	156	11.46%	17.2%	1363	4464	1187	$nD^{2.781}$

Table 2: Selected performances for general ideals (from DRL to LEX)

- (**Further impact**) The proposed probabilistic algorithm for ideals in shape position has been used to greatly improve computations involved in the *discrete logarithm problem* in [52], and the study on the structure of term ideals of generic systems and the sparsity of the multiplication matrix T_1 has led to a new complexity $O(n\log(D)D + \log(D)D^{\omega})$ for change of ordering for generic polynomial systems [51].

Effective algorithms for simple decomposition over finite fields

We propose several original algorithms for decomposing polynomial sets over finite fields into simple sets. There are two cases depending on whether the input polynomial set is zero-dimensional or not.

- (**Zero-dimensional case**) In this case the fields we are working on are algebraic extensions of finite fields and thus they are perfect. We generalize the squarefree decomposition approach for univariate polynomials over a finite field to that over

the field product determined by a simple set. In this generalization, pth root extraction in the field product is a key ingredient. An approach is proposed to compute the pth root by recursively solving linear equations. Based on this approach, we present algorithms to decompose any zero-dimensional polynomial set into simple sets over an arbitrary finite field. These algorithms can produce both ideal and zero decompositions of the given polynomial set.

- (**Positive-dimensional case**) In the positive-dimensional case, the key technique above is no longer applicable. By adjoining the parameters of the regular set to the finite ground field, the problem of simple decomposition over the ground field in the positive-dimensional case is then reduced to that over a purely transcendental extension of the ground field in the zero-dimensional case. We propose a method to solve the latter problem by using the properties of simple sets and the relationship we establish between simple sets and radical ideals. This method relies on the critical step of computing the radical of a special ideal over the field extension. Then we investigate the three existing methods developed by Matsumoto [106], Kemper [83], and Fortuna, Gianni, and Trager [62] for the computation of radicals over finite fields and, in particular, improve the FGT method (which is the most suitable) according to our specific setting to formulate a simplified and optimized algorithm.

- (**Experimental results**) The effectiveness of all these proposed algorithms is demonstrated by our preliminary experiments with implementations in Maple and Singular. In particular, comparisons on proposed algorithms in the zero-dimensional case and on the three methods for radical computation in the positive-dimensional case (see Table 3 below for selected examples) are also performed.

Ex	Char	FGT/Maple	Matsumoto/Maple	Kemper/Maple
Q4	3	0.516	104.047	205.421
Q7	3	81.609	>3600	720.015
Q11	17	0.203	>3600	206.407
Q18	53	7.11	460.921	>3600

Table 3: Selected timings for decomposing regular sets into simple sets (positive-dimensional)

Squarefree decomposition and factorization over unmixed products of field extensions

We first establish the isomorphism between unmixed products represented and quotient rings determined by simple sets and then study operations including arithmetic, gcd computation, squarefree decomposition, and factorization of polynomials over unmixed products by reducing them to polynomial computations modulo the saturated ideals of the simple sets. Some of the underlying ideas and techniques for our treatment have appeared in different forms in previous works on the theories and methods of triangular sets [5, 37, 74, 159], Gröbner bases [21, 50], and dynamic evaluation [42, 45, 16]. However, factorization of polynomials over unmixed products of field extensions is studied here likely for the first time.

Contributions in this direction include (1) identification of unmixed products of field extensions with representations by simple sets, (2) introduction of the concepts of squarefree decomposition, irreducibility, and factorization of polynomials over unmixed products, (3) algorithms for squarefree decomposition and factorization of polynomials over unmixed products, and (4) examples and experiments illustrating the algorithms proposed with an initial implementation.

Furthermore, one can easily design a new algorithm for simple decomposition based on the proposed algorithm for squarefree decomposition over unmixed products of field extensions. The designed algorithm will be free of computation of radicals of positive-dimensional ideals over finite fields.

Applications in Biology and Coding Theory

We have applied methods for solving polynomial systems and for efficient change of orderings of Gröbner bases over finite fields to practical problems arising in Biology and Coding Theory.

To be precise, two methods for polynomial system solving over finite fields, namely the traversal method based on Gröbner bases and a specialized method for Boolean triangular decomposition, have been applied to detect the steady states and their numbers for finite biological models like Boolean networks of the segment polarity genes in Drosophila melanogaster from [2, 87] and lac operon from [145]. This is a new application to use algebraic methods to study biological phenomena.

In another application to Coding Theory, we reduce the problem of finding an

interpolation polynomial which is minimal w.r.t. a term ordering in the Sudan list decoding algorithm to changing the term orderings of Gröbner bases of some ideals derived from the original problem. This reduction makes it possible to perform proposed algorithms for fast change of ordering of Gröbner bases to solve this particular problem in Coding Theory. We also briefly discuss the feasibility, difficulties and potential advantages of applying our algorithms to this problem. This is still ongoing work.

Perspectives

Following the research directions indicated in this thesis, some theoretical and practical problems remaining for further study are listed below.

- Deterministic algorithm for change of ordering using sparsity of multiplication matrices: The deterministic algorithm proposed in the thesis is only able to compute the LEX Gröbner bases of the radicals of given ideals by discarding the multiplicities of the solutions. Clearly the design of deterministic algorithms for change of ordering keeping the multiplicities is also important.

- Sparsity analysis for multiplication matrices other than T_1: The multiplication matrix T_1 is of special importance because it is related to the univariate polynomial in the LEX Gröbner basis of an ideal which contains most information of the solutions, and the sparsity analysis of T_1 is relatively easier to carry out, and thus in this thesis we mainly focus on T_1. But other multiplication matrices are also important, in particular in the general case where the BMS-based method is applicable.

- Algorithm for factorization over products of field extensions of finite fields of small characteristics. Designing such an algorithm has not be touched in this thesis, and the study on all the cases will be complete once corresponding algorithms are available. Potential study may proceed step by step with the zero-dimensional case first and then the positive-dimensional one, just like the study on algorithms for simple decompositions over finite fields.

- Application to the interpolation step in list decoding and new applications of

the algorithms proposed in the thesis, in particular those for fast change of orderings of Gröbner bases.

Organization of the thesis

In the first part of the thesis, the concepts, algorithms and usage of two mainstream tools for polynomial system solving, namely Gröbner bases and triangular sets, are reviewed respectively in Chapters 1 and 2. Chapter 3 furnishes additional background knowledge in commutative algebra and finite fields. The readers may refer to [35, 159] for more detailed descriptions of related theories.

The second part consists of main contributions of this thesis, presented in the same order as summarized above. Publications related to the results presented in this part include:

- Chapter 4: J.-C. Faugère and C. Mou. Fast algorithm for change of ordering of zero-dimensional Gröbner bases with sparse multiplication matrices. *Proceedings of ISSAC 2011*, 115–122, 2011
  J.-C. Faugère and C. Mou. Sparse FGLM algorithms. Preprint arXiv:1304.1238, 2013

- Chapter 5: X. Li, C. Mou, and D. Wang. Decomposing polynomial sets into simple sets over finite fields: The zero-dimensional case. *Computers and Mathematics with Applications*, 60: 2983–2997, 2010
  C. Mou, D. Wang, and X. Li. Decomposing polynomial sets into simple sets over finite fields: The positive-dimensional case. *Theoretical Computer Science*, 468: 102–113, 2013

- Chapter 6: C. Mou and D. Wang. Squarefree decomposition and factorization over unmixed products of field extensions. In preparation

- Section 7.1: X. Li, C. Mou, W. Niu, and D. Wang. Stability analysis for discrete biological models using algebraic methods. *Mathematics in Computer Science*, 5: 247–262, 2011

Part I

Background

Chapter 1

Gröbner bases

Gröbner bases have become a standard tool in the computational ideal theory, especially for solving polynomial systems. In this chapter we first recall basic preliminaries on multivariate polynomial rings, which are the common ground of the study in this thesis. Then we review the Buchberger algorithm for computing Gröbner bases and the FGLM algorithm for changing the term orderings. At the end of this chapter we focus on solving polynomial systems with Gröbner bases.

1.1 Preliminaries

In this thesis, we fix the notations $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$ as the ring of integers, and the fields of rational, real and complex numbers respectively. For a finite field $\mathbb{F}_q$, we always assume that it is a field of characteristic p with $q = p^m$ elements for some $m \in \mathbb{Z}$.

1.1.1 Multivariate polynomial ring

Let $\mathcal{R}$ be a commutative ring with unity, and $x_1, \ldots, x_n$ the indeterminates over $\mathcal{R}$. We fix the orderings of these indeterminates as $x_1 < \ldots < x_n$, and write $\boldsymbol{x}$ and $\boldsymbol{x}_i$ for short for $(x_1, \ldots, x_n)$ and $(x_1, \ldots, x_i)$ respectively.

Definition 1.1.1. For all $\alpha_i \geq 0$ $(i = 1, \ldots, n)$ with n an integer ≥ 1, the formal power product $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ is called a *term*, which may also be written as $\boldsymbol{x}^{\boldsymbol{\alpha}}$ for short with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$. A finite sum of terms $F = \sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}}$ with $c_{\boldsymbol{\alpha}} \in \mathcal{R}$ is called a

polynomial in $x_1, \ldots, x_n$ over $\mathcal{R}$. The element $c_\alpha \in \mathcal{R}$ is said to be the *coefficient* of F w.r.t. $\boldsymbol{x}^{\boldsymbol{\alpha}}$, and is denoted by $\mathrm{coef}(F, \boldsymbol{x}^{\boldsymbol{\alpha}})$.

For a term $\boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, α_i is called the *degree* of $\boldsymbol{x}^{\boldsymbol{\alpha}}$ w.r.t. x_i and denoted by $\deg(\boldsymbol{x}^{\boldsymbol{\alpha}}, x_i)$, and $\alpha_1 + \cdots + \alpha_n$ is called the *total degree* of $\boldsymbol{x}^{\boldsymbol{\alpha}}$, denoted by $\mathrm{tdeg}(\boldsymbol{x}^{\boldsymbol{\alpha}})$. Similarly, for a polynomial F,

$$\deg(F, x_i) = \max\{\deg(\boldsymbol{x}^{\boldsymbol{\alpha}}, x_i) : \mathrm{coef}(F, \boldsymbol{x}^{\boldsymbol{\alpha}}) \neq 0\}$$

is said to be the *degree* of F w.r.t. x_i, and

$$\mathrm{tdeg}(F) := \max\{\mathrm{tdeg}(\boldsymbol{x}^{\boldsymbol{\alpha}}) : \mathrm{coef}(F, \boldsymbol{x}^{\boldsymbol{\alpha}}) \neq 0\}$$

the *total degree* of F.

Defining additions and multiplications for any two polynomials in $x_1, \ldots, x_n$ over $\mathcal{R}$, one sees that the set of all polynomials in $x_1, \ldots, x_n$ over $\mathcal{R}$ forms a ring, and it is called the *polynomial ring* over $\mathcal{R}$ w.r.t. the *variables* $x_1, \ldots, x_n$. We write $\mathcal{R}[x_1, \ldots, x_n]$ or $\mathcal{R}[\boldsymbol{x}]$ for this polynomial ring. Furthermore, $\mathcal{R}[\boldsymbol{x}]$ is a *univariate* polynomial ring if $n = 1$, and a *multivariate* one otherwise, with polynomials within them also said to be *univariate* or *multivariate* respectively.

The polynomial ring over a field $\mathbb{K}$, in most cases the multivariate one, is the common ground this thesis will work on. A nonconstant polynomial F in $\mathbb{K}$ is called *irreducible* over $\mathbb{K}$ if it cannot be written as the product of two nonconstant polynomials. It should be noted that the irreducibility of a polynomial is dependent on the field we choose.

Furthermore, $\mathbb{K}[\boldsymbol{x}]$ is a unique factorization domain. In particular, if we write $F \in \mathbb{K}[\boldsymbol{x}]$ as $cF_1^{a_1} \cdots F_r^{a_r}$ for $c \in \mathbb{K}$, $a_i \in \mathbb{Z}^+$, and F_i monic, irreducible over $\mathbb{K}$ and pairwise distinct, we call it a *factorization* of F. In this case, $F_1 \cdots F_r$ is called the *squarefree part* of F. Furthermore, a polynomial G equal to its own squarefree part is called *squarefree*. This name is easier to understand with an equivalent definition that there does not exist $H \in \mathbb{K}[\boldsymbol{x}] \setminus \mathbb{K}$ such that $H^2 \mid G$.

1.1.2 Ideals and varieties

Ideals and varieties are two fundamental objects in Algebraic Geometry. With the development of effective tools for manipulate them like Gröbner bases, we are able to study them in a computational way.

Definition 1.1.2. Let $\mathcal{R}$ be a commutative ring. Then a subset $\mathfrak{a} \subset \mathcal{R}$ is said to be an *ideal* if

(a) $0 \in \mathfrak{a}$;

(b) if $F, G \in \mathfrak{a}$, then $F + G \in \mathfrak{a}$;

(c) for any $F \in \mathfrak{a}$ and $H \in \mathcal{R}$, we have $HF \in \mathfrak{a}$.

The subset $\langle F_1, \ldots, F_r \rangle := \{\sum_{i=1}^{r} H_i F_i : H_1, \ldots, H_r \in \mathcal{R}\}$ is easily verified to be an ideal by the definition, and we call it the *ideal generated by* $F_1, \ldots, F_r$. If a finite set of elements in $\mathcal{R}$ can be found to generate an ideal, we say that this ideal is *finitely generated*. One important fact about the multivariate polynomial ring $\mathbb{K}[\boldsymbol{x}]$ is that every ideal in it is finitely generated.

Theorem 1.1.1 (Hilbert Basis Theorem). *For any ideal* $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$*, there exist* $F_1, \ldots, F_r$ *such that* $\mathfrak{a} = \langle F_1, \ldots, F_r \rangle$.

In fact this theorem applies to all *Noetherian* rings, and one of the equivalent theorems to it is as follows.

Theorem 1.1.2 (Ascending Chain Condition). *For any ascending chain of ideals in* $\mathbb{K}[\boldsymbol{x}]$

$$\mathfrak{a}_1 \subset \mathfrak{a}_2 \subset \mathfrak{a}_3 \subset \cdots,$$

there exists $N \geq 1$ *such that*

$$\mathfrak{a}_N = \mathfrak{a}_{N+1} = \mathfrak{a}_{N+2} = \cdots.$$

The above theorem, also called *ascending chain condition*, is very useful to prove termination of algorithms related to ideal computations, for example the Buchberger algorithm in Section 1.2.

Definition 1.1.3. Let $F_1, \ldots, F_r$ be polynomials in $\mathbb{K}[\boldsymbol{x}]$. Then the set

$$\mathsf{V}(F_1, \ldots, F_r) := \{(a_1, \ldots, a_n) \in \mathbb{K}^n : F_i(a_1, \ldots, a_n) = 0 \ (i = 1, \ldots, r)\}$$

is called the *affine variety* defined by $F_1, \ldots, F_r$.

The word "affine" is used to differentiate the variety from the *projective variety*. In the case of no ambiguity, we omit "affine" for simplicity.

Let $\mathcal{F} = [F_1, \ldots, F_r]$ be a polynomial set in $\mathbb{K}[\boldsymbol{x}]$, and $\overline{\mathbb{K}}$ be the algebraic closure of $\mathbb{K}$. Then a point $\overline{\boldsymbol{x}} \in \overline{\mathbb{K}}$ is said to be a zero of $\mathcal{F}$ if $F_i(\overline{\boldsymbol{x}}) = 0$ for all $i = 1, \ldots, r$. A similar notion to the variety is the set of all the zeros of a polynomial set $\mathcal{F}$, denoted by $\mathrm{Zero}(\mathcal{F})$. The difference lies in the fact that each coordinate of the points in $\mathrm{Zero}(\mathcal{F})$ is from the algebraic closure $\overline{\mathbb{K}}$, but that in $\mathsf{V}(\mathcal{F})$ is from $\mathbb{K}$.

It is easy to prove that $\mathsf{V}(F_1, \ldots, F_r) = \mathsf{V}\langle F_1, \ldots, F_r\rangle$, and thus one can derive that a subset $V \subset \mathbb{K}^n$ is a variety if and only if an ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ can be found such that $V = \mathsf{V}(\mathfrak{a})$.

Definition 1.1.4. Let V be a subset of $\mathbb{K}^n$. Then the set

$$\mathsf{I}(V) = \{F \in \mathbb{K}[\boldsymbol{x}] : F(\boldsymbol{v}) = 0, \forall \boldsymbol{v} \in V\}$$

is an ideal, and we call it the *corresponding ideal* of V.

In fact, there exists one-one correspondence between varieties and a special kind of ideals called radical ones. We will explore this in more details in Section 3.1.2.

1.1.3 Term ordering

The term ordering is one important concept in the Gröbner basis theory, for it imposes an ordering on polynomials and polynomial sets and leads to termination criteria for algorithms related to Gröbner bases. We first define the term ordering on the set of all the terms w.r.t. $\boldsymbol{x}$, denoted by $\mathfrak{T}(\boldsymbol{x})$, and then extend it to $\mathcal{R}[\boldsymbol{x}]$.

Definition 1.1.5. A total order $<$ on $\mathfrak{T}(\boldsymbol{x})$ is said to be a *term ordering*, if

(a) for any term $\boldsymbol{u}_1, \boldsymbol{u}_2$ and $\boldsymbol{v} \in \mathfrak{T}(\boldsymbol{x})$, if $\boldsymbol{u}_1 < \boldsymbol{u}_2$, then $\boldsymbol{u}_1\boldsymbol{v} < \boldsymbol{u}_2\boldsymbol{v}$;

(b) $<$ is a well order, in other words, any nonempty subset of $\mathfrak{T}(\boldsymbol{x})$ has a minimal element.

For two terms $\boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ and $\boldsymbol{x}^{\boldsymbol{\beta}} = x_1^{\beta_1} \cdots x_n^{\beta_n} \in \mathfrak{T}(\boldsymbol{x})$, the following two term orderings are the most commonly used ones:

1. *Lexicographic ordering* (short as LEX): we say $\boldsymbol{x}^{\boldsymbol{\alpha}} <_{\mathsf{LEX}} \boldsymbol{x}^{\boldsymbol{\beta}}$ if there exists i $(1 \leq i \leq n)$ such that

$$\alpha_j = \beta_j \ (i+1 \leq j \leq n) \quad \text{and} \quad \alpha_i < \beta_i.$$

2. *Graded reverse lexicographic ordering* or *degree reverse lexicographic ordering* (short as DRL): we say $\boldsymbol{x}^{\boldsymbol{\alpha}} <_{\mathsf{DRL}} \boldsymbol{x}^{\boldsymbol{\beta}}$ if

$$\begin{aligned} \operatorname{tdeg}(\boldsymbol{x}^{\boldsymbol{\alpha}}) < \operatorname{tdeg}(\boldsymbol{x}^{\boldsymbol{\beta}}), \quad \text{or} \quad & \operatorname{tdeg}(\boldsymbol{x}^{\boldsymbol{\alpha}}) = \operatorname{tdeg}(\boldsymbol{x}^{\boldsymbol{\beta}}) \text{ and there exists} \\ & i \ (1 \leq i \leq n) \text{ such that} \\ & \alpha_j = \beta_j (1 \leq j \leq i-1) \text{ and } \alpha_i > \beta_i. \end{aligned}$$

Let $F \in \mathcal{R}[\boldsymbol{x}]$ be a nonzero polynomial, and $<$ a term ordering. Then the biggest term of F w.r.t. $<$ is called the *leading term* of F and denoted by $\operatorname{lt}_<(F)$. The coefficient of F w.r.t. $\operatorname{lt}_<(F)$ is the *leading coefficient* of F and denoted by $\operatorname{lc}_<(F)$. When no ambiguity occurs, we can write $\operatorname{lt}(F)$ and $\operatorname{lc}(F)$ for simplicity. In the case of $\operatorname{lc}(F) = 1$, we call F a *monic* polynomial.

Any term ordering $<$ on $\mathfrak{T}(\boldsymbol{x})$ also induces an ordering $<'$ on $\mathcal{R}[\boldsymbol{x}]$ in the following recursive way:

(a) For any nonzero polynomial $F \in \mathcal{R}[\boldsymbol{x}]$, $0 <' F$;

(b) For any $F, G \in \mathcal{R}[\boldsymbol{x}]$ nonzero, $F <' G$ if and only if

$$\operatorname{lt}(F) < \operatorname{lt}(G) \quad \text{or} \quad \operatorname{lt}(F) = \operatorname{lt}(G) \text{ and } F - \operatorname{lt}(F) <' G - \operatorname{lt}(G).$$

We also denote the induced ordering on $\mathcal{R}[\boldsymbol{x}]$ by $<$ for simplicity.

1.1.4 Polynomial reduction

Next we discuss a kind of reduction on polynomials in $\mathbb{R}[\boldsymbol{x}]$, where terms of polynomials are of our interest. In the following we restrict to the simple case where $\mathcal{R}$ is simply a field $\mathbb{K}$.

Definition 1.1.6. Fix a term ordering $<$. For any polynomials $F, P \in \mathbb{K}[\boldsymbol{x}]$ with $P \neq 0$, if there exist a term $\boldsymbol{u}$ of F and $\boldsymbol{v} \in \mathfrak{T}(\boldsymbol{x})$ such that $\boldsymbol{u} = \boldsymbol{v}\operatorname{lt}(F)$, then we say F is *reducible* by P. In this case, let

$$G = F - \frac{\operatorname{coef}(F, \boldsymbol{u})}{\operatorname{lc}(P)} \cdot \boldsymbol{u}P.$$

Then we say F is reduced to G by P by eliminating $\boldsymbol{u}$, and denote this process by $F \xrightarrow[\boldsymbol{u}]{P} G$, simplified as $F \xrightarrow{P} G$ if the term $\boldsymbol{u}$ is not of our interest.

For a polynomial set $\mathcal{P} \subset \mathbb{K}[\boldsymbol{x}]$, if there exists $P \in \mathcal{P}$ such that $F \xrightarrow[\boldsymbol{u}]{P} G$, we say F is reduced to G by $\mathcal{P}$, and denote this process by $F \xrightarrow[\boldsymbol{u}]{\mathcal{P}} G$. In this case F is said to be *reducible* by $\mathcal{P}$; otherwise it is *reduced*. Suppose after a number of reductions on F by $\mathcal{P}$ the resulting polynomial R becomes reduced. Then we call R the *normal form* of F by $\mathcal{P}$, denote it by $\mathrm{nform}(F, \mathcal{G})$.

We can perform the reduction of a polynomial by a finite set of polynomials with the following algorithm.

Algorithm 1: $([Q_1, \ldots, Q_s], R) := \mathsf{PolyRed}([P_1, \ldots, P_s], F)$

Input: $[P_1, \ldots, P_s] \subseteq \mathbb{K}[\boldsymbol{x}], F \in \mathbb{K}[\boldsymbol{x}]$.

Output: $[Q_1, \ldots, Q_s] \subseteq \mathbb{K}[\boldsymbol{x}], R \in \mathbb{K}[\boldsymbol{x}]$ such that

(a) $F = \sum_{i=1}^{s} Q_i P_i + R$;

(b) R is reduced by $\{P_1, \ldots, P_s\}$;

(c) If $Q_i P_i \neq 0$, then $\mathrm{lt}(Q_i P_i) \leq \mathrm{lt}(F)$.

1.1 $Q_i := 0$ $(i = 1, \ldots, s), R := F$;
1.2 **while** R is reducible by $\{P_1, \ldots, P_s\}$ **do**
1.3 Choose P_i such that R is reducible by P_i;
1.4 Choose a monomial λ such that $R \xrightarrow{P_i} R - \lambda P_i$;
1.5 $R := R - \lambda P_i$;
1.6 $Q_i := Q_i + \lambda$;
1.7 **end**
1.8 **return** $([Q_1, \ldots, Q_s], R)$;

According to the requirements of the algorithm on the output R, obviously we have $F - R \in \langle P_1, \ldots, P_s \rangle$. If the computed normal form $R = 0$, then $F \in \langle P_1, \ldots, P_s \rangle$. Therefore, it is hopeful to test ideal membership by this algorithm. But two major drawbacks of this reduction algorithm are as follows:

(a) The normal form is not unique. It depends on the choice of P_i at Line 1.3;

(b) $R = 0$ is only a sufficient condition for $F \in \langle P_1, \ldots, P_s \rangle$, but not necessary.

This reduction process is also one of the fundamental operations on polynomials in the theory of Gröbner bases. And we will see that only when combined with Gröbner bases can this algorithm overcome the drawbacks and become much more powerful.

1.2 Gröbner bases and Buchberger algorithm

Definition 1.2.1. For a set $S \subset \mathbb{N}^n$, we call the ideal generated by the set of terms $\{\boldsymbol{x}^{\boldsymbol{\alpha}} \in \boldsymbol{\alpha} \in S\}$ the *term ideal* of S, and denote it by $\langle \boldsymbol{x}^{\boldsymbol{\alpha}} : \boldsymbol{\alpha} \in S \rangle$.

Proposition 1.2.1. *For a term ordering and an ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$, we have*

(a) $\langle \mathrm{lt}(\mathfrak{a}) \rangle$ is a term ideal;

(b) there exist $G_1, \ldots, G_s \subset \mathbb{K}[\boldsymbol{x}]$ such that $\langle \mathrm{lt}(\mathfrak{a}) \rangle = \langle \mathrm{lt}(G_1) \ldots, \mathrm{lt}(G_s) \rangle$.

Definition 1.2.2. Fix a term ordering $<$ and an ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$. If a finite subset $\mathcal{G} = \{G_1, \ldots, G_s\}$ of $\mathfrak{a}$ satisfies

$$\langle \mathrm{lt}(G_1), \ldots, \mathrm{lt}(G_s) \rangle = \langle \mathrm{lt}(\mathfrak{a}) \rangle,$$

then we call G a *Gröbner basis* of $\mathfrak{a}$ w.r.t. $<$.

It can be proved that a Gröbner basis $\mathcal{G}$ of an ideal $\mathfrak{a}$ is also a set of generators of it, namely $\langle \mathcal{G} \rangle = \mathfrak{a}$. When the term ordering is clear, one may omit it when referring to a Gröbner basis. Furthermore, when referring to a Gröbner basis $\mathcal{G}$, if the ideal $\mathfrak{a}$ is not explicitly stated, then we mean that $\mathcal{G}$ is the Gröbner basis of $\langle \mathcal{G} \rangle$.

The term ordering plays an important role in the theory of Gröbner bases. For example, from the definition one may expect Gröbner bases w.r.t. different term orderings are also different, for the term ideal $\langle \mathrm{lt}(\mathfrak{a}) \rangle$ is dependent on the chosen term ordering. One main contribution of the thesis is on change of term orderings of Gröbner bases, and we will further study this issue in Chapter 4.

The following theorem lists several equivalent properties for a polynomial set to be a Gröbner basis.

Theorem 1.2.2. *Fix a term ordering. Then for an ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ and a subset $\mathcal{G}$ of $\mathfrak{a}$, the following statements are equivalent.*

(a) $\mathcal{G}$ *is a Gröbner basis of* $\mathfrak{a}$.

(b) For any polynomial $F \in \mathfrak{a}$*, there exists* G_i *such that* $\mathrm{lt}(G_i) \mid \mathrm{lt}(F)$.

(c) For any polynomial $F \in \mathbb{K}[\boldsymbol{x}]$*,* $\mathrm{nform}(F, \mathcal{G})$ *is unique.*

(d) For any polynomial $F \in \mathfrak{a}$*,* $\mathrm{nform}(F, \mathcal{G}) = 0$*;*

As indicated by Items (c) and (d) of Theorem 1.2.2, one can see that the Gröbner basis, as a special set of generators for an ideal, overcomes the two drawbacks summarized in Section 1.1.4 for ordinary polynomial reductions. Therefore, we can claim that the problem of ideal membership test can be solved algorithmically once the Gröbner basis of the ideal is given.

Gröbner bases w.r.t. LEX are of the best algebraic structure and convenient to use for many practical problems. One of its remarkable properties is related to elimination of variables. Fix the LEX ordering on $\mathbb{K}[\boldsymbol{x}]$ with $x_1 < \cdots < x_n$. For any l $(1 \leq l \leq n)$ and ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$, one can show that $\mathfrak{a} \cap \mathbb{K}[\boldsymbol{x}_l]$ is still an ideal, and we call it the lth *elimination ideal* of $\mathfrak{a}$, denoted by $\mathfrak{a}_l$.

Theorem 1.2.3 (Elimination theorem). *Let* $\mathcal{G} \subset \mathbb{K}[\boldsymbol{x}]$ *be a Gröbner basis of an ideal* $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ *w.r.t. LEX. Then the polynomial set* $\mathcal{G} \cap \mathbb{K}[\boldsymbol{x}_l]$ *is still a Gröbner basis of* $\mathfrak{a}_l$ *w.r.t. LEX.*

Combined with the Extension theorem with Gröbner bases, the Elimination theorem furnishes the foundation of solving polynomial systems. We will touch this issue later in Section 1.4.4.

A Gröbner basis can be further processed so that it is unique for the ideal. Such Gröbner bases are called *reduced* ones. In fact in most Computer Algebra systems the output Gröbner bases are reduced so that the user may compare them from various systems.

Definition 1.2.3. A Gröbner basis $\mathcal{G}$ is called a *reduced Gröbner basis* if

(a) Each polynomial in $\mathcal{G}$ is monic.

(b) For any $G \in \mathcal{G}$, G is reduced modulo $\mathcal{G} \setminus \{G\}$.

Proposition 1.2.4. *Any ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ has a unique reduced Gröbner basis w.r.t. a given term ordering.*

The uniqueness of reduced Gröbner bases not only provide a method for comparison of different output Gröbner bases of various algorithms, but also make it possible to test the ideal equality: two ideals $\langle F_1, \ldots, F_s \rangle$ and $\langle G_1, \ldots, G_r \rangle$ in $\mathbb{K}[\boldsymbol{x}]$ are the same if and only if their reduced Gröbner bases w.r.t. some term ordering coincide.

Now it is clear that many problems related to ideals are reduced to the computation of their Gröbner bases. Along with the introduction of Gröbner bases, Buchberger also proposed an algorithm for computing them. Nowadays this algorithm is called the *Buchberger algorithm* in general.

One key object in the Buchberger algorithm is the S-polynomial. For two terms $\boldsymbol{u} = x_1^{k_1} \cdots x_n^{k_n}$ and $\boldsymbol{v} = x_1^{l_1} \cdots x_n^{l_n}$, it is easy to see that $\mathrm{lcm}(\boldsymbol{u}, \boldsymbol{v}) = x_1^{m_1} \cdots x_n^{m_n}$, where $m_i = \max(k_i, l_i)$.

Definition 1.2.4. For two nonzero polynomials $F, G \in \mathbb{K}[\boldsymbol{x}]$, let $\boldsymbol{u} = \mathrm{lcm}(\mathrm{lt}(F), \mathrm{lt}(G))$. Then the polynomial

$$\mathrm{S}(F, G) = \mathrm{lc}(G) \cdot \frac{\boldsymbol{u}}{\mathrm{lt}(F)} \cdot F - \mathrm{lc}(F) \cdot \frac{\boldsymbol{u}}{\mathrm{lt}(G)} \cdot G$$

is called the *S-polynomial* of F and G.

The term "S" in the S-polynomial derives from syzygies, a concept from Algebraic Geometry. From the definition of S-polynomials, one can see that in the S-polynomial of F and G, in fact the leading terms of F and G are canceled. The Buchberger algorithm is based on a necessary and sufficient condition for a polynomial set to be a Gröbner basis, which can be easily turned into an algorithm. This following condition is called the *Buchberger's criterion*, or *S-pair criterion*.

Theorem 1.2.5. *Let $\mathcal{G} = \{G_1, \ldots, G_s\}$ be a generator set of an ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$. Then $\mathcal{G}$ is a Gröbner basis of $\mathfrak{a}$ if and only if for any $i \neq j$, $\mathrm{nform}(\mathrm{S}(G_i, G_j), \mathcal{G}) = 0$.*

With this criterion it is natural to have the Buchberger algorithm (Algorithm 2).

Algorithm 2: Buchberger algorithm $\mathcal{G} := \mathsf{GröbnerBasis}(\mathcal{F})$

Input: $\mathcal{F} = [F_1, \ldots, F_s] \subseteq \mathbb{K}[\boldsymbol{x}]$.

Output: A Gröbner basis $\mathcal{G}$ of $\langle \mathcal{F} \rangle$ such that $\mathcal{F} \subseteq \mathcal{G}$.

```
2.1  G := F;
2.2  L := {{G_i, G_j} : G_i, G_j ∈ G such that G_i ≠ G_j};
2.3  while L ≠ ∅ do
2.4      {G_i, G_j} := pop(L);
2.5      R := a normal form of S(G_i, G_j) modulo G;
2.6      if R ≠ 0 then
2.7          L := L ∪ {{G, R} : G ∈ G};
2.8          G := G ∪ {R};
2.9      end
2.10 end
2.11 return G;
```

Combining the above algorithm with two other criteria proposed also by Buchberger for speeding up (see e.g., [14, Section 5.5]), we have a relatively efficient algorithm for computing Gröbner bases. This makes Gröbner bases a computational tool to study problems of ideals, and now we complete the algorithmic method for several such problems, such as the tests for ideal membership and ideal equality.

1.3 FGLM algorithm

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. LEX have good algebraic structures and are convenient to use for polynomial system solving, while those w.r.t. DRL are computationally easy to obtain. Therefore, as shown in Figure 1, the common strategy to solve a polynomial system via Gröbner bases is to first compute the Gröbner basis of the ideal defined by the system w.r.t. DRL, and then change its ordering to LEX.

In this section we review the FGLM algorithm, which performs the change of ordering of Gröbner bases of zero-dimensional ideals efficiently [53]. One main feature of this well-known algorithm is the close link exploited in it between the change of ordering and linear algebra operations in the quotient ring.

Suppose that $\mathcal{G}_1$ is the Gröbner basis of a zero-dimensional ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ w.r.t. a term ordering $<_1$. Now one aims at computing the Gröbner basis $\mathcal{G}_2$ of $\mathfrak{a}$ w.r.t. another ordering $<_2$. Denote by D the degree of $\mathfrak{a}$ (the dimension of $\mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$ as a vector space). The FGLM algorithm consists of three major steps as follows.

(1) One computes the canonical basis of $\mathbb{K}[\boldsymbol{x}]/\langle \mathcal{G}_1 \rangle$ and orders its elements following $<_1$. Let $B = [\epsilon_1, \ldots, \epsilon_D]$ be the ordered basis. Since $<_1$ is a term ordering, one knows that $\epsilon_1 = \mathbf{1}$ for sure.

(2) For any variable x_i $(i = 1, \ldots, n)$, one can compute $\mathrm{nform}(\epsilon_j x_i, \mathcal{G}_1)$ for $j = 1, \ldots, D$, which can be viewed as an element in $\mathbb{K}[\boldsymbol{x}]/\langle \mathcal{G}_1 \rangle$ and thus can also be written as a linear combination of B. Taking the coefficients of this combination as the 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 $\mathfrak{a}$ for x_i. The multiplication matrix for x_i indeed represents the linear map $\phi_i : B \to B$ defined by $\phi_i(\epsilon) = \mathrm{nform}(x_i \epsilon, \mathcal{G}_1)$. It is not hard to verify that all T_i commute: $T_i T_j = T_j T_i$ for $i, j = 1, \ldots, n$.

(3) Then the terms in $\mathbb{K}[\boldsymbol{x}]$ are handled one by one w.r.t. $<_2$. 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 FGLM guarantee that once a linear dependency of the coordinate vectors of computed terms

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

is found, a polynomial $F \in \mathcal{G}_2$ can be directly derived in the following form

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

where $\boldsymbol{x}^{\boldsymbol{l}} = \mathrm{lt}(F)$ w.r.t $<_2$ [53].

To obtain the whole Gröbner basis $\mathcal{G}_2$, all one needs to do is to compute the coordinate vector of each term one by one, and check whether a linear dependency of these vectors occurs once a new vector is computed (which can be realized by maintaining an echelon form of the matrix constructed from the computed vectors).

A trivial upper bound for the number of terms to consider is $D + 1$ because of the vector size.

As for the complexity for the FGLM algorithm to perform the change of ordering, the computational cost to compute all the multiplication matrices are $O(nD^3)$ arithmetic operations [53, Proposition 3.1], and the whole construction in Step (3) also needs $O(nD^3)$ arithmetic operations (obtained by analyzing the cost of maintaining the echelon matrix). To summarize, the change of ordering for zero-dimensional Gröbner bases can be completed with $O(nD^3)$ operations in total with the FGLM algorithm [53, Proposition 4.1].

As one can see, all operations in these three steps are merely matrix manipulations from linear algebra. That is why the FGLM algorithm is efficient for change of ordering. In this algorithm, the multiplication matrices play an important role as the bridge between the change of ordering and linear operations. In Chapter 4 we will further extend the FGLM algorithm to have more efficient variants which take advantage of the sparsity of multiplication matrices.

1.4 Gröbner bases for polynomial system solving

The general method to solve a polynomial equation set is to transform it into several equation sets of triangular-like shapes with the same solutions, and then extend the solution of each univariate polynomial within to that of the whole polynomial equation set. Therefore we start the study on polynomial system solving with Gröbner bases with univariate polynomial equations.

1.4.1 Univariate polynomial equation solving

For a univariate polynomial $F \in \mathbb{K}[x]$, now consider its solutions in some algebraic extension $\tilde{\mathbb{K}}$ of $\mathbb{K}$. By the algebraic fundamental theorem, F has at most $\deg(F)$ roots in $\tilde{\mathbb{K}}$, and they are the solutions of the equation $F = 0$. Therefore by solving $F = 0$ we mean to find an appropriate way to represent all these roots in $\tilde{\mathbb{K}}$ of F.

First, to obtain the solutions of $F = 0$ in $\mathbb{Q}$ one can use factorization of F over $\mathbb{Q}$. Take $F = 2x^4 - 3x^3 + 2x^2 - 1$ for example. With methods for factorization (see

e.g., [153, Chapters 14&15]), one can factorize F as

$$F = (x-1)(2x+1)(x^2-x+1).$$

Then it is known that there exist two and only two rational solutions of $F = 0$: $x = 1$ and $x = 1/2$. Take the fact that all integers and rational numbers can be represented precisely in computers, one knows that the whole solving process of $F = 0$ in $\mathbb{Q}$ can be implemented in computers.

Then for the solutions in $\mathbb{R}$ and $\mathbb{C}$, we can use formula by radicals to represent precisely the solutions of $F = 0$ if $\deg(F) \leq 4$. However, the solvability by radicals has been proved to be not feasible for general equations of degree ≥ 5 as in the Galois theory [86, Chapter VI]. But this does not mean that we can do nothing to solve equations with higher degrees, for example we can adopt the numeric methods like the Newton method [82].

But in Computer Algebra we have a symbolic method to represent the real solutions of a polynomial equation, that is the real root isolation [30, 136]. It means to compute a sequence of disjoint rational intervals such that each real root of F is contained in some interval and there is only one root in each interval. As the lengths of the intervals can be arbitrarily small, we can regard these intervals as a "precise" representation of all the real roots of F without errors. Or in other words, all the distinct real solutions can be computed precisely.

To summarize, the solutions of a univariate polynomial equation in various number fields can be represented in an appropriate way, and there exist applicable methods to solve them. Therefore, in the following we always assume that any univariate polynomial equation is solvable.

1.4.2 Number of solutions of polynomial equation sets

For a polynomial equation set in $\mathbb{K}[\boldsymbol{x}]$

$$F_1(\boldsymbol{x}) = 0, \quad \ldots, \quad F_s(\boldsymbol{x}) = 0,$$

we call $\mathcal{F} = \{F_1, \ldots, F_s\}$ its *defining polynomial set*. It is already shown that $\mathrm{Zero}(\mathcal{F}) = \mathrm{Zero}(\langle F \rangle)$, and thus the solutions of a polynomial equation set are uniquely determined by the ideal generated by its defining polynomial set. Then the study on solving $\mathcal{F} = 0$ is reduced to that on the corresponding ideal $\langle \mathcal{F} \rangle$.

Suppose that $\mathcal{F} = 0$ has some solution in $\overline{\mathbb{K}}$. If the number of solutions if finite, then $\mathcal{F}$ is said to be *zero-dimensional*, otherwise *positive-dimensional*. The following results show how to determine whether a given polynomial set $\mathcal{F} = 0$ has a solution and whether $\mathcal{F}$ is zero-dimensional.

Theorem 1.4.1 (Hilbert's Weak Nullstellensatz). *Let $\mathbb{K}$ be an algebraically closed field and $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ be an ideal. Then $V(\mathfrak{a}) = \emptyset$ if and only if $1 \in \mathfrak{a}$.*

By the above theorem, $\mathcal{F} = 0$ has no solutions if and only if $1 \in \langle F \rangle$. We can determine whether $1 \in \langle F \rangle$ by computing the reduced Gröbner bases of $\langle \mathcal{F} \rangle$, and thus it can be decided whether $\mathcal{F} = 0$ has a solution.

Theorem 1.4.2. *Let $\mathcal{G}$ be a Gröbner basis of $\mathcal{F} \subset \mathbb{K}[\boldsymbol{x}]$ w.r.t. any term ordering. Then $\mathcal{F}$ is zero-dimensional if and only if for any $i = 1, \ldots, n$, there exists a positive integer m_i and polynomial $G_i \in \mathcal{G}$ such that $\mathrm{lt}(G_i) = x_i^{m_i}$.*

Example 1.4.1. For the Cyclic 3 system

$$\begin{cases} x_1 + x_2 + x_3 = 0, \\ x_1x_2 + x_2x_3 + +x_3x_1 = 0, \\ x_1x_2x_3 - 1 = 0, \end{cases}$$

the Gröbner basis of the ideal generated by its defining polynomial set w.r.t. LEX is $\{x_1^3 - 1, x_2^2 + x_1x_2 + x_1^2, x_3 + x_2 + x_1\}$, and thus one knows that this polynomial set is zero-dimensional. Similarly, one can confirm that Cyclic 5 is also zero-dimensional while Cyclic 4 is not by computing the corresponding Gröbner bases.

1.4.3 Zero-dimensional polynomial equation set

In previous discussions we have made clear that the common method to solve a zero-dimensional polynomial equation set via Gröbner bases consists of three steps: computing Gröbner bases w.r.t. DRL, changing their orderings to LEX, and transforming Gröbner bases w.r.t. LEX to either triangular sets or rational univariate representations. We have reviewed the first two steps in this chapter, next we briefly show how the rational univariate representation (short as RUR) can be used to represent the solutions of zero-dimensional polynomial equation sets, leaving the contents for transformation to triangular sets in Section 2.4.

The rational univariate representation is a special representation of the solutions of zero-dimensional polynomial equation sets over $\mathbb{Q}$ (in fact it is effective over fields of characteristic 0 and of "large" positive characteristics) [135]. For a polynomial equation set defined by $\mathcal{F} \subset \mathbb{Q}[\boldsymbol{x}]$, its RUR is composed of two parts: (1) a *separating variable*, which is usually a linear combination of $x_1, \ldots, x_n$; (2) a polynomial set in $\mathbb{Q}[x_0, x_1, \ldots, x_n]$ of the shape

$$\begin{cases} H(x_0) = 0, \\ x_1 = \dfrac{G_1(x_0)}{G(x_0)}, \ldots, x_n = \dfrac{G_n(x_0)}{G(x_0)}. \end{cases}$$

The RUR for $\mathcal{F} = 0$ has the following properties.

(a) The solutions of $H(x_0) = 0$ correspond to the solutions of $\mathcal{F} = 0$, and so are the multiplicities of these solutions.

(b) All the solutions of $\mathcal{F} = 0$ can be computed by substituting the solutions of $H(x_0) = 0$ into other equations.

(c) Almost all (with only finitely many exceptions) linear combinations of $x_1, \ldots, x_n$ can be chosen as the separating variable.

(d) Once the separating variable is chosen, the rational univariate representation exists and is unique.

From the above properties of RUR one sees that solving $\mathcal{F} = 0$ is almost equivalent to solving the univariate polynomial equation $H(x_0) = 0$ in the RUR, for the coordinates $x_1, \ldots, x_n$ of its solution can be directly computed by Property (b). Therefore it can be concluded that RUR is a good representation of zero-dimensional polynomial equation sets over $\mathbb{Q}$. But it should also be mentioned that RUR only works for fields of characteristic 0.

As for computation of RUR, an effective method has been proposed for transforming the LEX Gröbner basis into RUR [135]. This completes the whole process of solving zero-dimensional polynomial equation sets via Gröbner bases and RUR.

Example 1.4.2. It is easy to verify that

$$\begin{cases} 2\,x^3 - 3\,xy + 4 = 0, \\ 3\,x^2 y - y^2 + x - 1 = 0 \end{cases}$$

is zero-dimensional, and its rational univariate representation is

$$\begin{cases} 14\,t^6 + 29\,t^3 - 9\,t^2 - 16 = 0, \\ x = \dfrac{-29\,t^3 + 12\,t^2 + 32}{28\,t^5 + 29\,t^2 - 6\,t}, \\ y = \dfrac{18\,t^4 + 8\,t^3 + 60\,t - 8}{28\,t^5 + 29\,t^2 - 6\,t}, \end{cases}$$

where t is the introduced separating variable.

1.4.4 Positive-dimensional polynomial equation set

Now suppose that $\mathcal{F} \subset \mathbb{K}[\boldsymbol{x}]$ is a positive-dimensional polynomial equation set, and $\mathcal{G}$ is a Gröbner basis of $\mathfrak{a} = \mathcal{F}$ w.r.t. LEX. Then by the Elimination Theorem (Theorem 1.2.3), one knows that $\mathcal{G} \cap \mathbb{K}[\boldsymbol{x}_l]$ is precisely a Gröbner basis of the elimination ideal $\mathfrak{a}_l$.

Let $\boldsymbol{a} = (a_1, \ldots, a_l) \in \mathrm{Zero}(\mathfrak{a}_l)$. Then we call $\boldsymbol{a}$ a *partial solution* of the polynomial equation set $\mathcal{F} = 0$. The following theorem furnishes a sufficient condition to extend a partial solution.

Theorem 1.4.3 (Extension Theorem). *Let $\mathfrak{a} = \langle F_1, \ldots, F_s \rangle$ be an ideal in $\mathbb{C}[\boldsymbol{x}]$, and $\mathfrak{a}_{n-1}$ be its $(n-1)$th elimination ideal. For each $i = 1, \ldots, s$, write F_i in the form*

$$F_i = G_i x_n^{N_i} + H_i,$$

where $N_i \geq 0$, $G_i \in \mathbb{C}[\boldsymbol{x}_{n-1}]$ nonzero, and $\deg(H_i, x_n) < N_i$. Suppose that $\boldsymbol{c} = (c_1, \ldots, c_{n-1}) \in \mathrm{Zero}(\mathfrak{a}_{n-1})$ is a partial solution. If $\boldsymbol{c} \notin \mathrm{Zero}(\{G_1, \ldots, G_s\})$, then there exists $c_n \in \mathbb{C}$ such that $(\boldsymbol{c}, c_n) \in \mathrm{Zero}(\mathfrak{a})$.

It is easy to prove that for any ideal $\mathfrak{a}$, $\mathfrak{a}_l$ is the lth elimination ideal of $\mathfrak{a}_{l+1}$. Therefore the Extension Theorem above can be applied step by step, as long as the condition within is satisfied, until the solution of the polynomial equation set is computed. This is the general way to solve polynomial equation systems including positive-dimensional ones.

Corollary 1.4.4. *Suppose that $\mathfrak{a} = \langle F_1, \ldots, F_s \rangle \subset \mathbb{C}[\boldsymbol{x}]$ is an ideal, and there exist i $(1 \leq i \leq s)$ such that F_i is in the form*

$$F_i = c x_n^N + H_i,$$

where $N_i \leq 0$, $c \in \mathbb{C}$ *is a constant, and* $\deg(H_i, x_n) < N$. *If* $\boldsymbol{c} = (c_1, \ldots, c_{n-1}) \in \mathrm{Zero}(\mathfrak{a}_{n-1})$ *is a partial solution, then there exists* $c_n \in \mathbb{C}$ *such that* $(\boldsymbol{c}, c_n) \in \mathrm{Zero}(\mathfrak{a})$.

Chapter 2

Triangular sets

After the introduction of characteristic sets by J.F. Ritt, triangular sets have evolved into an alternative tool for manipulating ideals besides Gröbner bases. In this chapter, we introduce basic concepts and terminologies related to triangular sets, and review the Wu–Ritt algorithm for computing characteristic sets and several common triangular sets with different properties like regular, simple and irreducible triangular sets. Similarly, at the end the way how to use triangular sets to solve polynomial systems is discussed.

2.1 Concepts and terminologies

Similar to the term ordering in the Gröbner basis theory, in the theory of triangular sets one inherent ordering is endowed to all the polynomials in $\mathcal{R}[\boldsymbol{x}]$ based on the fixed ordering on all the variables $x_1 < \ldots < x_n$.

For any variable x_i $(1 \leq i \leq n)$, one has $\mathcal{R}[\boldsymbol{x}] = \mathcal{R}[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n][x_i]$. The biggest variable x_p appearing in a polynomial $F \in \mathcal{R}[\boldsymbol{x}]$ is called the *leading variable* of F, denoted by $\mathrm{lv}(F)$. Suppose $\mathrm{lv}(F) = x_k$. Then F can be written as $F = Ix_p^d + R$, where $\deg(R, x_p) < d$. We call I, R and d here the *initial*, *tail* and *leading degree* of F, and denote them by $\mathrm{ini}(F)$, $\mathrm{tail}(F)$ and $\mathrm{ldeg}(F)$ respectively.

These terms are mainly used in the triangular set theory, which essentially views polynomials in $\mathcal{R}[\boldsymbol{x}]$ in an inductive way on the variables $x_1, \ldots, x_n$. Following this point of view, a polynomial reduction is defined as follows.

Proposition 2.1.1. *Let F, G be two polynomials in $\mathcal{R}[\boldsymbol{x}]$, and x_k a variable. Suppose that $m = \deg(F, x_k)$ and $l = \deg(G, x_k)$. If $l > 0$, then there exist $Q, R \in \mathcal{R}[\boldsymbol{x}]$ and s $(0 \leq s \leq m - l + 1)$ such that*

$$\mathrm{lc}(G, x_k)^s F = QG + R \quad \textit{and} \quad \deg(R, x_k) < l. \tag{2.1}$$

Furthermore, if s is fixed then Q and R are unique.

The process to compute Q and R from F and G is called the *pseudo division* of F to G w.r.t. x_k, and the polynomials Q and R are the *pseudo quotient* and *pseudo remainder* of F to G w.r.t. x_k respectively, denoted by $\mathrm{prem}(F, G, x_k)$ and $\mathrm{pquo}(F, G, x_k)$. The equation in (2.1) is said to be the *pseudo division formula*. In particular, a polynomial $P \in \mathbb{K}[\boldsymbol{x}]$ is said to be *reduced* w.r.t. G if $\deg(P, \mathrm{lv}(G)) < \mathrm{ldeg}(G)$. Apparently the pseudo remainder $\mathrm{prem}(P, G, \mathrm{lv}(G))$ is reduced w.r.t. G.

The following algorithm performs the pseudo division of F to G w.r.t. x_k to compute the pseudo quotient Q and pseudo remainder R.

Algorithm 3: $(Q, R, s) := \mathsf{Prem}(F, G, x_k)$

Input: Polynomials $F, G \in \mathbb{R}[\boldsymbol{x}]$, and a variable x_k such that $\deg(G, x_k) > 0$.

Output: $Q = \mathrm{pquo}(F, G, x_k)$, $R = \mathrm{prem}(F, G, x_k)$, and s as in (2.1)

3.1 $R := F$; $Q := 0$; $l := \deg(G, x_k)$; $s := 0$;
3.2 **while** $\deg(R, x_k) \geq l$ **do**
3.3 $\quad r := \deg(R, x_k)$;
3.4 $\quad R := \mathrm{lc}(G, x_k)R - \mathrm{lc}(R, x_k)x_k^{r-l}G$;
3.5 $\quad Q := \mathrm{lc}(G, x_k)Q + \mathrm{lc}(R, x_k)x_k^{r-l}$;
3.6 $\quad s := s + 1$;
3.7 **end**
3.8 **return** (Q, R, s);

The pseudo division is an important operation on polynomials in $\mathcal{R}[\boldsymbol{x}]$. It is similar to divisions of univariate polynomials over a field, but in $\mathcal{R}[x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n]$ the inverse of an element does not necessarily exist, and thus we have to multiply $\mathrm{lc}(G, x_k)$ as in (2.1). Given two polynomials F and G, with the pseudo division we are able to product a new polynomial R whose degree w.r.t. the fixed variable is smaller, and a certain zero relation between these polynomials hold (see [159] for details). The process from F to R via G can be viewed as a reduction which focuses on a specific

variable of the polynomials. This reduction process plays a fundamental role in the theory of triangular sets, and will be used extensively in related algorithms.

Definition 2.1.1. An ordered polynomial set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}] \setminus \mathbb{K}$ is said to be a *triangular set* if $\mathrm{lv}(T_1) < \cdots < \mathrm{lv}(T_r)$.

Clearly the truncated polynomial set $[T_1, \ldots, T_i]$ for $i = 1, \ldots, r$ is also a triangular set, and we denote it by $\mathcal{T}_{\leq i}$. The variables $\mathrm{lv}(T_1), \ldots, \mathrm{lv}(T_r)$ are called the *dependents* of $\mathcal{T}$, while the others called its *parameters*. In the case when $r = n$, a triangular set is *zero-dimensional*; otherwise *positive-dimensional* (Note that these definitions coincide with those for polynomial sets).

For example, the ordered polynomial set $\mathcal{T} = [ux_1^2+3x_1+5, (x_1-1)x_2^2+ux_1x_2+4]$ is a triangular set in $\mathbb{Q}[u, x_1, x_2]$. The dependents of $\mathcal{T}$ are x_1 and x_2, and the parameter is u. It is obvious that $\mathcal{T}$ is positive-dimensional.

For a triangular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$, rename all its dependents as $y_i = \mathrm{lv}(T_i)$ for $i = 1, \ldots, r$ and its parameters as $u_1, \ldots, u_{n-r}$. Then by the definition of triangular sets, one sees that $\mathcal{T}$ is still a triangular set for the variable ordering $u_1 < \cdots < u_{n-1} < y_1 < \cdots < y_r$. Without loss of generality, we always assume such an ordering of the variables once a triangular set is given, that is, the parameters are ordered smaller than the dependents. We write $\boldsymbol{u}$ and $\boldsymbol{y}$ for $(u_1, \ldots, u_{n-r})$ and $(y_1, \ldots, y_r)$ for short, and $\boldsymbol{y}_i$ for $(y_1, \ldots, y_i)$ similarly.

A triangular set $\mathcal{T}$ is of a triangular form w.r.t. their leading variables as follows:

$$\mathcal{T} = \begin{bmatrix} T_1(x_1, \ldots, x_{p_1}) \\ T_2(x_1, \ldots, x_{p_1}, \ldots, x_{p_2}) \\ \vdots \\ T_r(x_1, \ldots, x_{p_1}, \ldots, x_{p_2}, \ldots, x_{p_r}) \end{bmatrix}, \tag{2.2}$$

where $\mathrm{lv}(T_i) = x_{p_i}$ for $i = 1, \ldots, r$, and $p_1 < p_2 < \cdots < p_r \leq n$.

This form makes the solutions of $\mathcal{T} = 0$ for a triangular set $\mathcal{T}$ are easy to compute. One only needs to solve $T_1 = 0$ to find a solution $\overline{\boldsymbol{x}}_1$, then substitute $\overline{\boldsymbol{x}}_1$ to T_2 to solve $T_2(\overline{\boldsymbol{x}}_1, \ldots, x_{p_2})$ for a solution $\overline{\boldsymbol{x}}_2$. Repeating this process will lead to the full solution of $\mathcal{T} = 0$. This is particularly the case when $\mathcal{T}$ is zero-dimensional, for in this case each time we only need to solve a univariate polynomial equation, which is much easier.

Now we are able to extend the pseudo division to that w.r.t. a triangular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$, and this is possible because the leading variables of the polynomials in $\mathcal{T}$ are distinct. The pseudo remainder of a polynomial $F \in \mathbb{K}[\boldsymbol{x}]$ w.r.t. $\mathcal{T}$ is recursively defined by

$$\operatorname{prem}(F, \mathcal{T}) = \operatorname{prem}(\operatorname{prem}(F, T_r), \mathcal{T}_{\leq r-1}),$$

where $\mathcal{T}_{\leq r-1} = [T_1, \ldots, T_{r-1}]$ is the truncated triangular set. For a polynomial set $\mathcal{P} \subset \mathbb{K}[\boldsymbol{x}]$, $\operatorname{prem}(\mathcal{P}, \mathcal{T}) := \{\operatorname{prem}(P, \mathcal{T}) : P \in \mathcal{P}\}$. And a polynomial $F \in \mathbb{K}[\boldsymbol{x}]$ is said to be *reduced* w.r.t. $\mathcal{T}$ if F is reduced w.r.t. every T_i in $\mathcal{T}$ for $i = 1, \ldots, r$. One can easily see that $\operatorname{prem}(F, \mathcal{T})$ is reduced w.r.t. $\mathcal{T}$.

Applying the pseudo division formula (2.1) inductively, we can also have the following formula for pseudo division w.r.t. a triangular set.

$$(\prod_{i=1}^{r} \operatorname{ini}(T_i)^{d_i})P = \sum_{i=1}^{r} Q_i T_i + \operatorname{prem}(P, \mathcal{T}), \tag{2.3}$$

where d_i are integers, and $Q_j \in \mathbb{K}[\boldsymbol{x}]$ for $i = 1, \ldots, r$. This formula furnishes a zero relationship as follows. For two polynomial sets $\mathcal{P}, \mathcal{Q} \subset \mathbb{K}[\boldsymbol{x}]$, we denote the set $\operatorname{Zero}(\mathcal{P}) \setminus \operatorname{Zero}(\mathcal{Q})$ by $\operatorname{Zero}(\mathcal{P}/\mathcal{Q})$.

Proposition 2.1.2. *For any polynomial $P \in \mathbb{K}[\boldsymbol{x}]$ and triangular set $\mathcal{T} \subset \mathbb{K}[\boldsymbol{x}]$, if $\operatorname{prem}(P, \mathcal{T}) = 0$, then*

$$\operatorname{Zero}(\mathcal{T}/\operatorname{ini}(\mathcal{T})) \subset \operatorname{Zero}(P),$$

where $\operatorname{ini}(\mathcal{T}) = \{\operatorname{ini}(T) : T \in \mathcal{T}\}$.

This zero relationship is the starting point of characteristic sets, one kind of important triangular sets in their development. This relationship also addresses the importance of one kind of ideals of our interest for triangular sets: the *saturated ideal* of a triangular set $\mathcal{T} = [T_1, \ldots, T_r]$ is defined as

$$\operatorname{sat}(\mathcal{T}) = \langle \mathcal{T} \rangle : H^{\infty} = \{G : \exists s \geq 0 \text{ such that } H^s G \in \langle \mathcal{T} \rangle\}$$

with $H = \prod_{i=1}^{r} \operatorname{ini}(T_i)$. For the sake of brevity, we write $\operatorname{sat}_i(\mathcal{T}) := \operatorname{sat}([T_1, \ldots, T_i])$.

2.2 Characteristic sets and Wu–Ritt algorithm

Definition 2.2.1. A triangular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$ is said to be an *ascending set*, if each polynomial T_i is reduced w.r.t. all T_j $(j \neq i)$ for $i = 1, \ldots, r$. Furthermore, for a polynomial set $\mathcal{P} \subset \mathbb{K}[\boldsymbol{x}]$, an ascending set $\mathcal{C}$ is a *characteristic set* of $\mathcal{P}$ if $\mathcal{C} \subset \langle \mathcal{P} \rangle$, and $\operatorname{prem}(\mathcal{P}, \mathcal{C}) = \{0\}$.

As one can see from the definition of characteristic sets and Proposition 2.1.2, the notion of characteristic sets is introduced to study the zeros of polynomial sets. The following zero relationship makes a step further.

Proposition 2.2.1. *Let $\mathcal{T} = [T_1, \ldots, T_r]$ be a characteristic set of a polynomial set $\mathcal{P} \subset \mathbb{K}[\boldsymbol{x}]$. Denote $\mathcal{P}_i = \mathcal{P} \cup \{\operatorname{ini}(T_i)\}$ for $i = 1, \ldots, r$. Then*

$$\begin{aligned} &\operatorname{Zero}(\mathcal{T}/\operatorname{ini}(\mathcal{T})) \subset \operatorname{Zero}(\mathcal{P}) \subset \operatorname{Zero}(\mathcal{T}), \\ &\operatorname{Zero}(\mathcal{T}/\operatorname{ini}(\mathcal{T})) = \operatorname{Zero}(\mathcal{P}/\operatorname{ini}(\mathcal{T})), \\ &\operatorname{Zero}(\mathcal{P}) = \operatorname{Zero}(\mathcal{T}/\operatorname{ini}(\mathcal{T})) \cup \bigcup_{i=1}^{r} \operatorname{Zero}(\mathcal{P}_i). \end{aligned} \tag{2.4}$$

Next we recall the Wu–Ritt algorithm for computing a characteristic set of a given polynomial set, which is a typical algorithm for triangular decompositions.

We first define an ordering on polynomials and then extend it to triangular sets. This ordering is crucial in the termination of the Wu–Ritt algorithm.

Definition 2.2.2. For two non-zero polynomials $F, G \in \mathbb{K}[\boldsymbol{x}]$, F is said to have *a lower rank* than G if either $\operatorname{lv}(F) < \operatorname{lv}(G)$, or $\operatorname{lv}(F) = \operatorname{lv}(G) > 0$ and $\operatorname{ldeg}(F) < \operatorname{ldeg}(G)$. In this case, G is said to have *a higher rank* than F. We denote this rank between F and G by $F \prec G$ or $G \succ F$.

If neither $F \prec G$ nor $G \prec F$, then F and G are said to have *the same rank*, denoted by $F \sim G$.

Definition 2.2.3. For two triangular sets $\mathcal{T} = [T_1, \ldots, T_r]$ and $\mathcal{P} = [P_1, \ldots, P_s]$, $\mathcal{T}$ is said to have *a lower rank* than $\mathcal{P}$ if either (a) or (b) below holds.

(a) There exists $j < \min(r, s)$ such that

$$T_1 \sim S_1, \ldots, T_{j-1} \sim S_{j-1}, T_j \prec S_j;$$

(b) $r < s$, and

$$T_1 \sim S_1, \ldots, T_r \sim S_r.$$

In this case, $\mathcal{P}$ is said to have *a higher rank* than $\mathcal{T}$.

If neither $\mathcal{T} \prec \mathcal{S}$ nor $\mathcal{S} \prec \mathcal{T}$, then $\mathcal{T}$ and $\mathcal{S}$ are said to have *the same rank*, denoted by $\mathcal{T} \sim \mathcal{S}$.

In the above definition, if $\mathcal{T} \sim \mathcal{S}$, then $r = s$, and $T_1 \sim S_1, \ldots, T_r \sim S_r$. One sees that the rank $\preceq$ between two triangular sets is indeed a partial ordering on all the triangular sets. The following theorem plays the same role in the termination of the Wu–Ritt algorithm as the Ascending Chain Condition (Theorem 1.1.2) in the Buchberger algorithm.

Theorem 2.2.2. *Let*

$$\mathcal{T}_1 \succeq \mathcal{T}_2 \succeq \cdots \succeq \mathcal{T}_k \succeq \cdots$$

be a sequence of triangular sets with ranks never getting higher. Then there exists an integer k' such that $\mathcal{T}_k \sim \mathcal{T}_{k'}$ for all $k > k'$.

With the partial ordering $\preceq$, given a set of ascending sets we can talk about the "minimal" one (if it exists). For a polynomial set $\mathcal{F}$, let $\mathbb{P}$ be the set of all ascending sets contained in $\mathcal{F}$. Then clearly $\mathbb{P} \neq \emptyset$ because the set consisting of one single polynomial is also an ascending set. We call the minimal ascending set in $\mathbb{P}$ the *basic set* of $\mathcal{F}$. Existence of the basic set has been proved, and one can compute it from $\mathcal{F}$ with the Algorithm 4.

Algorithm 4: Basic set algorithm $\mathcal{B} := \text{BasicSet}(\mathcal{F})$

Input: $\mathcal{F} \subseteq \mathbb{K}[\boldsymbol{x}]$.

Output: $\mathcal{B}$, a basic set of $\mathcal{F}$.

4.1 $\mathcal{P} := \mathcal{F}$, $\mathcal{B} := \emptyset$;
4.2 **while** $\mathcal{P} \neq \emptyset$ **do**
4.3 $B :=$ an element in $\mathcal{P}$ with the lowest rank;
4.4 $\mathcal{B} := \mathcal{B} \cup [B]$;
4.5 **if** $\text{tdeg}(B) = 0$ **then**
4.6 $\mathcal{P} := \emptyset$;
4.7 **else**
4.8 $\mathcal{P} := \{P \in \mathcal{P} \setminus \{B\} : P \text{ is reduced w.r.t. } B\}$;
4.9 **end**
4.10 **end**
4.11 **return** $\mathcal{B}$;

With Algorithm 4, the Wu–Ritt algorithm (Algorithm 5) to compute characteristic sets from polynomial sets is natural.

Algorithm 5: Wu–Ritt algorithm $\mathcal{C} := \text{CharSet}(\mathcal{F})$

Input: $\mathcal{F} \subseteq \mathbb{K}[\boldsymbol{x}]$.

Output: $\mathcal{B}$, a characteristic set of $\mathcal{F}$.

5.1 $\mathcal{P} := \mathcal{F}$, $\mathcal{R} := \mathcal{F}$;
5.2 **while** $\mathcal{R} \neq \emptyset$ **do**
5.3 $\mathcal{C} := \text{BasicSet}(\mathcal{P})$;
5.4 **if** $\mathcal{C}$ *is contradictory* **then**
5.5 $\mathcal{R} := \emptyset$;
5.6 **else**
5.7 $\mathcal{R} := \text{prem}(\mathcal{F} \setminus \mathcal{C}, \mathcal{C})$;
5.8 $\mathcal{P} := \mathcal{P} \cup \mathcal{R}$;
5.9 **end**
5.10 **end**
5.11 **return** $\mathcal{C}$;

Based on the zero relation stated in Proposition 2.2.1 (in particular the last equation), the Wu–Ritt algorithm can be extended to compute finite characteristic sets

$\mathcal{T}_1, \ldots, \mathcal{T}_s$ such that

$$\mathrm{Zero}(\mathcal{F}) = \bigcup_{i=1}^{s} \mathrm{Zero}(\mathcal{T}_i / \mathrm{ini}(\mathcal{T}_i))$$

for a polynomial set $\mathcal{F}$. The relation above turns the study on $\mathrm{Zero}(\mathcal{F})$ totally to that on the zeros of each $\mathcal{T}_i$ and $\mathrm{ini}(\mathcal{T}_i)$. As stated previously, the zeros of triangular sets are relatively easier to compute, and thus we can conclude that this approach is a feasible method for solving polynomial systems.

With the ability to study zeros of polynomial sets entitled by the characteristic set, Wu applied his algorithm to automatic proving of elementary geometric theorems [164, 165]. We sketch his method by as follows.

Step 1: Turn an elementary geometric theorem into the corresponding algebraic form via Descartian coordinates. For example, the geometric relationship for two segments to be parallel can be described by the gradients of them with their coordinates. In fact, most elementary geometric relationship can be expressed in an algebraic way. The hypothesis of a theorem is turned into a polynomial equation set $\mathcal{H} = 0$, while the conclusion is a polynomial equation $G = 0$.

Step 2: Compute an appropriate characteristic set $\mathcal{T}$ of $\mathcal{H}$.

Step 3: Compute the pseudo remainder $R = \mathrm{prem}(G, \mathcal{T})$. If $R \equiv 0$, then we know the theorem holds under the condition $\mathrm{ini}(\mathcal{T}) \neq 0$.

This method due to Wu contains insights on the relationship between geometry and algebra. After a rewriting of the geometric theorem in the algebraic form, the characteristic set provides a tool to test the zero relation. What worths mentioning is that the condition $\mathrm{ini}(\mathcal{T}) \neq 0$ indeed reflects the degenerative conditions for the geometric objects.

2.3 Regular, simple and irreducible triangular sets

2.3.1 Regular sets

Let $\mathcal{R}$ be a commutative ring with identity and $\mathfrak{a} \subset \mathcal{R}$ an ideal . An element $p \in \mathcal{R}$ is said to be *regular in* $\mathcal{R}$ if p is neither zero nor a zero divisor in $\mathcal{R}$, and *regular modulo* $\mathfrak{a}$ if its standard image in $\mathcal{R}/\mathfrak{a}$ is regular.

Definition 2.3.1. A triangular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$ is called a *regular set* if $\mathrm{ini}(T_i)$ is regular modulo $\mathrm{sat}_{i-1}(\mathcal{T})$ for $i = 2, \ldots, r$.

Regular sets are also known as regular chains introduced by Kalkbrener [77] and as proper ascending chains due to Yang and Zhang [167]. They were generalized by Wang [158] to regular systems.

The regular set is one kind of commonly used triangular sets. Next we list some of its remarkable properties for later use. Recall that $\boldsymbol{u}$ and $\boldsymbol{y}$ are the parameters and dependents of a triangular set respectively.

Proposition 2.3.1 ([5, Theorem 6.1], [158, Theorem 5.1]). *For any regular set $\mathcal{T} \subseteq \mathbb{K}[\boldsymbol{x}]$ and polynomial $P \in \mathbb{K}[\boldsymbol{x}]$,*

(a) $P \in \mathrm{sat}(\mathcal{T})$ if and only if $\mathrm{prem}(P, \mathcal{T}) = 0$;

(b) P is regular modulo $\mathrm{sat}(\mathcal{T})$ if and only if $res(P, \mathcal{T}) \neq 0$, where $res(P, \mathcal{T})$ is the recursively defined resultant of P w.r.t. $\mathcal{T}$.

Proposition 2.3.2 ([74, Theorem 4.4]). *Let $\mathcal{T}$ be a regular set in $\mathbb{K}[\boldsymbol{x}]$. Then*

(a) $\mathrm{sat}(\mathcal{T}) \neq \langle 1 \rangle$;

(b) $\mathrm{sat}(\mathcal{T})$ is unmixed-dimensional and its parameters form a transcendental basis of every associated prime of $\mathrm{sat}(\mathcal{T})$.

Proposition 2.3.3 ([74, Proposition 5.18]). *For any zero-dimensional regular set $\mathcal{T} \subseteq \mathbb{K}[\boldsymbol{x}]$, $\mathrm{sat}(\mathcal{T}) = \langle \mathcal{T} \rangle$.*

Proposition 2.3.4 ([74, Proposition 5.8]). *Let $\mathcal{T}$ be a regular set in $\mathbb{K}[\boldsymbol{x}]$. Then $\mathcal{T}_{\leq i}$ is a regular set in $\mathbb{K}[\boldsymbol{u}][\boldsymbol{y}_i]$ and for any dependent y_i of $\mathcal{T}$,*

(a) $\mathrm{sat}_i(\mathcal{T}) = \mathrm{sat}(\mathcal{T}) \cap \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}_i]$;

(b) the associated primes of $\mathrm{sat}_i(\mathcal{T})$ are the intersections of the associated primes of $\mathrm{sat}(\mathcal{T})$ with $\mathbb{K}[\boldsymbol{u}][\boldsymbol{y}_i]$.

Another kind of triangular sets similar to the regular set is the normal set: $\mathcal{T} \subset \mathbb{K}[\boldsymbol{x}]$ is called a *normal triangular set* if each $\mathrm{ini}(T_i)$ is a nonzero polynomial in $\mathbb{K}[\boldsymbol{u}]$. One can transform any regular set $\mathcal{T}$ into a normal triangular set $\mathcal{N}$, such that $\mathrm{sat}(\mathcal{N}) = \mathrm{sat}(\mathcal{T})$ [159, 94].

By decomposition of a polynomial set $\mathcal{F}$ into triangular sets we mean to compute finite triangular sets $\mathcal{T}_1, \ldots, \mathcal{T}_s$ such that $\mathrm{Zero}(\mathcal{F}) = \bigcup_{i=1}^{s} \mathrm{Zero}(\mathcal{T}_i / \mathrm{ini}(\mathcal{T}))$, or algebraically $\sqrt{\mathcal{F}} = \bigcap_{i=1}^{s} \mathrm{sat}(\mathcal{T}_i)$.

As the above relations show, to design an algorithm for triangular decomposition, we need to split somewhere in the algorithm so as to turn one polynomial set into several triangular ones. In addition, different triangular sets with different properties impose different conditions on the splitting technique in corresponding algorithms. As for algorithms for regular decomposition (i.e., the triangular sets in the decomposition is regular), the key technique used for splitting is the following pgcd algorithm.

Specification 6: $\{(G_1, \mathcal{A}_1), \ldots, (G_s, \mathcal{A}_s)\} := \mathsf{pgcd}(\mathcal{F}, \mathcal{T})$

Input: $\mathcal{T}$ — a regular set in $\mathbb{K}[\boldsymbol{x}]$, where $\mathbb{K}$ is an arbitrary field;
$\mathcal{F}$ — a polynomial set in $\mathbb{K}[\boldsymbol{x}][z]$.

Output: $\{(G_1, \mathcal{A}_1), \ldots, (G_s, \mathcal{A}_s)\}$ — a set of pairs such that

(a) each $\mathcal{A}_i$ is a regular set in $\mathbb{K}[\boldsymbol{x}]$, and $\mathrm{sat}(\mathcal{T}) \subseteq \mathrm{sat}(\mathcal{A}_i)$;

(b) $\sqrt{\mathrm{sat}(\mathcal{T})} = \sqrt{\mathrm{sat}(\mathcal{A}_1)} \cap \cdots \cap \sqrt{\mathrm{sat}(\mathcal{A}_s)}$;

(c) $\langle \mathcal{F} \rangle = \langle G_i \rangle$ in $\mathrm{fr}(\mathbb{K}[\boldsymbol{x}] / \mathrm{sat}(\mathcal{A}_i))[z]$;

(d) $G_i \in \langle \mathcal{F} \rangle + \mathrm{sat}(\mathcal{A}_i)$;

(e) $G_i = 0$, or $\mathrm{ini}(G_i)$ is regular modulo $\mathrm{sat}(\mathcal{A}_i)$.

In the algorithm description, for a ring $\mathcal{R}$, $\mathrm{fr}(\mathcal{R})$ denotes the total quotient ring of $\mathcal{R}$. In other words, it is the localization of $\mathcal{R}$ at the multiplicatively closed set of all its non-zerodivisors. A remarkable behavior of pgcd is that the splitting in it occurs only when it is necessary, and this splitting property is indeed derived from the well-known D5 principle [42, 16], which is introduced to perform the case discussion dynamically.

2.3.2 Simple and irreducible sets

Simple sets are also called squarefree regular sets [74]. From this name one also expect that simple sets are regular sets with some squarefree properties. One of the main contributions of this thesis is the study on algorithms for simple decompositions over finite fields. Here we only review the basic definitions of simple sets, leaving detailed

discussions to Chapter 5.

Definition 2.3.2. For a regular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$ with $\mathrm{lv}(T_i) = x_{p_i}$ for $i = 1, \ldots, r$, a point $\boldsymbol{\xi}$ is called a *regular zero* of $\mathcal{T}$ if for each $i = 1, \ldots, r$, $T_i(\boldsymbol{\xi}) = 0$, and $\boldsymbol{\xi}$ is of the form

$$\boldsymbol{\xi} = (x_1, \ldots, x_{p_1-1}, \xi_{p_1}, x_{p_1+1}, \ldots, x_{p_r-1}, \xi_{p_r}, x_{p_r+1} \ldots, x_n) \in \tilde{\mathbb{K}}^n,$$

where $\tilde{\mathbb{K}}$ is the algebraic closure of the field generated by all the parameters x_k as transcendental elements over $\mathbb{K}$. We denote the set of all regular zeros of $\mathcal{T}$ by $\mathrm{RZero}(\mathcal{T})$.

A regular zero of $\mathcal{T}$ indeed corresponds to a finitely generated field extensions of $\mathbb{K}$, where the variables with constrains from polynomials of $\mathcal{T}$ are algebraic elements and the others transcendental ones. The introduction of regular zeros reduces the study on regular sets to that on an irreducible components (i.e. a field extension from the algebraic object defined by the regular set). This idea will be further exploited in Chapter 5 in the study on simple sets.

Definition 2.3.3. A regular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$ with $\mathrm{lv}(T_i) = x_{p_i}$ for $i = 1, \ldots, r$ is called a *simple set* if for $i = 1, \ldots, r-1$ and regular zero $\overline{\boldsymbol{x}}_i \in \tilde{\mathbb{K}}^i$ of $\mathcal{T}_i = [T_1, \ldots, T_i] \subset \mathbb{K}[x_1, \ldots, x_{p_{i+1}-1}]$, where $\tilde{\mathbb{K}}$ is the algebraic closure of the field generated by all the parameters of $\mathcal{T}$ as transcendental elements over $\mathbb{K}$, the univariate $T_{i+1}(\overline{\boldsymbol{x}}_i, x_{p_{i+1}})$ w.r.t. $x_{p_{i+1}}$ is squarefree.

As explained earlier, a regular zero corresponds to a finitely generated field extension. In the theory of triangular sets, we also have one kind of triangular sets which are directly corresponding to finitely generated field extensions. The following notion of irreducible sets is recursively defined.

Definition 2.3.4. A triangular set in the form $[T] \subset \mathbb{K}[\boldsymbol{x}]$ is said to be an *irreducible set* if T is irreducible over $\mathbb{K}$. Then a regular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$ with $\mathrm{lv}(T_i) = x_{p_i}$ for $i = 1, \ldots, r$ is said to be an *irreducible set* if for any $i = 2, \ldots, r$, $[T_1, \ldots, T_{i-1}]$ is an irreducible set, and for any regular zero $\overline{\boldsymbol{x}}_{i-1} \in \tilde{\mathbb{K}}^{i-1}$ of $[T_1, \ldots, T_{i-1}] \subset \mathbb{K}[x_1, \ldots, x_{p_i-1}]$, where $\tilde{\mathbb{K}}$ is the algebraic closure of the field generated by all the parameters of $\mathcal{T}$ as transcendental elements over $\mathbb{K}$, $T_i(\overline{\boldsymbol{x}}_{i-1}, x_{p_i})$ as a univariate polynomial in x_{p_i} is irreducible.

In fact the triangular set is exactly the defining polynomial set of a finitely generated field extension. For transcendental elements $\alpha_1, \ldots, \alpha_s$ of $\mathbb{K}$, denote the transcendental extension $\mathbb{K}(\alpha_1, \ldots, \alpha_s)$ of $\mathbb{K}$ by $\tilde{\mathbb{K}}$. Suppose β_i $(i = 1, \ldots, r, s+r = n)$ are algebraic over $\tilde{\mathbb{K}}$, with $F_i(x_{s+i})$ the minimal polynomial of β_i over $\tilde{\mathbb{K}}(\beta_1, \ldots, \beta_{i-1})$. Then we have the *defining polynomial set* $F = \{F_1, \ldots, F_r\} \subset \mathbb{K}[\boldsymbol{x}]$ of the field extension $\mathbb{K}(\alpha_1, \ldots, \alpha_s, \beta_1, \ldots, \beta_r)$ over $\mathbb{K}$.

One can also derive the following correspondence between irreducible sets and finitely generated field extensions. We include the proof for the convenience of later use in Chapter 6.

Proposition 2.3.5. *An irreducible set determines a finitely generated field extensions, and vice versa.*

Proof. Given an irreducible set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$, rewrite it as in $\mathbb{K}[\boldsymbol{u}][\boldsymbol{y}]$. Then $\mathbb{K}(\boldsymbol{u})$ is a transcendental extension of $\mathbb{K}$. By the definition of irreducible sets, for each $i = 1, \ldots, r$, T_i is irreducible over $\mathbb{K}(\boldsymbol{u})(\beta_1, \ldots, \beta_{i-1})$, where β_j is defined by $T_j(\beta_1, \ldots, \beta_{j-1})$ as its minimal polynomial (perhaps transformation into monic one by division is necessary). Then $\mathbb{K}(\boldsymbol{u})(\beta_1, \ldots, \beta_r)$ is the desired finitely generated field extension of $\mathbb{K}$.

Suppose there exists a finitely generated field extension $\mathbb{K}(\boldsymbol{u})(\beta_1, \ldots, \beta_r)$ of $\mathbb{K}$. Then for each $i = 1, \ldots, r$, β_i is determined by a minimal polynomial P_i over $\mathbb{K}(\boldsymbol{u})(\beta_1, \ldots, \beta_{i-1})$. Replacing all β_j in P_i by y_j and multiplying the lcm of all denominators of P_i, we will have the irreducible set we want. □

2.4 Triangular sets for polynomial system solving

In the case when the characteristic of the base field is 0, with regular sets one is also able to test whether a given polynomial equation set is zero-dimensional or not.

Theorem 2.4.1. *Let* $\mathrm{char}(\mathbb{K}) = 0$, $\mathcal{F} \subset \mathbb{K}[\boldsymbol{x}]$, *and* $\mathcal{T}_1, \ldots, \mathcal{T}_s$ *be the regular sets in the regular decomposition of* $\mathcal{F}$. *Then* $\mathcal{F}$ *is zero-dimensional if and only if* $|\mathcal{T}_i| = n$ *for* $i = 1, \ldots, s$.

As the consequence of Theorem 2.4.1, given a polynomial equation set $\mathcal{F} = 0$, we can first compute its regular decomposition $\mathcal{T}_1, \ldots, \mathcal{T}_s$ to check whether it is zero-dimensional. If so, then for $i = 1, \ldots, s$, $|\mathcal{T}_i| = n$. That is to say, the polynomial set

defined by $\mathcal{T}_i$ is of the form

$$\begin{cases} T_{i,1}(x_1) = 0, \\ T_{i,2}(x_1, x_2) = 0, \\ \quad \cdots\cdots \\ T_{i,n}(x_1, \ldots, x_n) = 0, \end{cases}$$

and for any $\bar{\boldsymbol{x}} \in \mathrm{Zero}(T_1, \ldots, T_{i-1})$, $\mathrm{ini}(T_i)(\bar{\boldsymbol{x}}) \neq 0$ holds. To compute $\mathrm{Zero}(\mathcal{T}_i)$ one needs to solve the univariate polynomial $T_{i,1}(x_1) = 0$ with any applicable method (see Section 1.4.1), substitute its solution into $T_{i,2}(x_1, x_2)$ and solve the resulting univariate equation, and repeat this process until all the solutions of $\mathcal{T}_i$ are obtained. In this way, all the solutions of the zero-dimensional $\mathcal{F} = 0$ can be explicitly computed.

Example 2.4.1. Consider the following polynomial equation set in $\mathbb{Q}[x_1, \ldots, x_4]$

$$\begin{cases} x_2x_3 - 1 = 0, \\ x_4^2 + x_1x_2x_3 = 0, \\ x_1x_2x_4 + x_3^2 - x_2 = 0, \\ x_1x_3x_4 - x_3 + x_2^2 = 0. \end{cases} \tag{2.5}$$

With the variable ordering $x_1 < \cdots < x_4$ we can compute the regular sets of its defining polynomial set as

$$\mathcal{C}_1 = [x_1^3 + 4, x_2^3 + 1, x_2x_3 - 1, 2\,x_4 + x_1^2], \quad \mathcal{C}_2 = [x_1, x_2^3 - 1, x_2x_3 - 1, x_4].$$

Then by Theorem 2.4.1 we know the equation set (2.5) is zero-dimensional.

With recursive solving and substituting of the polynomial equation set defined by either regular set, we can obtain all the solutions of (2.5) as

$$\begin{array}{ccc} (0, 1, 1, 0), & (0, -\alpha, -\boldsymbol{\beta}, 0), & (0, -\boldsymbol{\beta}, -\alpha, 0), \\ (-\gamma, -1, -1, -\gamma^2/2), & (-\gamma, \alpha, \boldsymbol{\beta}, -\gamma^2/2), & (-\gamma, \boldsymbol{\beta}, \alpha, -\gamma^2/2), \\ (\alpha\gamma, -1, -1, \boldsymbol{\beta}\gamma^2/2), & (\alpha\gamma, \alpha, \boldsymbol{\beta}, \boldsymbol{\beta}\gamma^2/2), & (\alpha\gamma, \boldsymbol{\beta}, \alpha, \boldsymbol{\beta}\gamma^2/2), \\ (\boldsymbol{\beta}\gamma, -1, -1, \alpha\gamma^2/2), & (\boldsymbol{\beta}\gamma, \alpha, \boldsymbol{\beta}, \alpha\gamma^2/2), & (\boldsymbol{\beta}\gamma, \boldsymbol{\beta}, \alpha, \alpha\gamma^2/2), \end{array}$$

where

$$\alpha = \frac{1 - \sqrt{-3}}{2}, \quad \boldsymbol{\beta} = \frac{1 + \sqrt{-3}}{2}, \quad \gamma = \sqrt[3]{4}.$$

A triangular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$ with $\mathrm{lv}(T_i) = x_{p_i}$ is said to have the *projection property* if for each $i = 1, \ldots, r$ and any zero $\overline{\boldsymbol{x}} \in \mathrm{RZero}(\mathcal{T}_{\leq i})$, there exist $\overline{x}_{p_{i+1}}, \ldots, \overline{x}_{p_r}$ such that $(\overline{\boldsymbol{x}}, x_{p_i+1}, \ldots, x_{p_{i+1}-1}, \overline{x}_{p_{i+1}}, \ldots, \overline{x}_n) \in \mathrm{RZero}(\mathcal{T})$. It can be proved that regular, simple and irreducible sets all have the projection property.

The projection property is necessary for using triangular sets to solve positive-dimensional polynomial systems, for with this property we know each partial regular zero (in this case with parameters) can be extended to the complete regular zero. And therefore the process to solve positive-dimensional polynomial systems is to decompose them into triangular sets with the projection property, and solve the polynomials in all obtained triangular sets one by one.

Besides the rational univariate representation which is mainly over $\mathbb{Q}$, the triangular set is another good representation for solutions of polynomial equation sets. As one important step for solving polynomial systems, we present the algorithm to transform Gröbner bases w.r.t. LEX to finitely many triangular sets (indeed irreducible ones) [91]. The key techniques used in the algorithm below are operations and factorization over finitely generated field extensions [113, 160].

Algorithm 7: Gröbner bases to triangular sets $\mathbb{T} := \mathsf{LexTriangular}(\mathcal{G})$

Input: $\mathcal{G}$, Gröbner basis of $\mathcal{F} \subseteq \mathbb{K}[\boldsymbol{x}]$ w.r.t. LEX, with $\mathcal{F}$ zero-dimensional.

Output: $\mathbb{T}$, irreducible decomposition of $\mathcal{F}$.

7.1 $\mathbb{T} := \{\emptyset\}$;
7.2 **for** $i = 1, \ldots, n$ **do**
7.3 $\quad \mathcal{H} := \{G \in \mathcal{G} : \mathrm{lv}(G) = x_i\}$;
7.4 $\quad$ Reorder $\mathcal{H}$ according to the polynomial ordering;
7.5 $\quad \mathbb{U} := \mathbb{T}$; $\mathbb{T} := \{\emptyset\}$;
7.6 $\quad$ **for** $\mathcal{M} \in \mathbb{U}$ **do**
7.7 $\quad\quad$ **repeat**
7.8 $\quad\quad\quad P :=$ the first element in $\mathcal{H}$; $\mathcal{H} := \mathcal{H} \setminus P$; $Q := \mathrm{ini}(P)$;
7.9 $\quad\quad\quad Q :=$ the inverse of Q in $\mathbb{K}[x_1, \ldots, x_{i-1}]/\langle \mathcal{M} \rangle$;
7.10 $\quad\quad$ **until** $Q \neq 0$;
7.11 $\quad$ **end**
7.12 $\quad$ Factorize P into irreducible factors $Q_1, \ldots, Q_r$ in $\mathbb{K}[x_1, \ldots, x_{i-1}]/\langle \mathcal{M} \rangle$;
7.13 $\quad \mathbb{T} := \mathbb{T} \cup \{\mathcal{M} \cup \{Q_i\} : i = 1, \ldots, r\}$;
7.14 **end**
7.15 **return** $\mathbb{T}$

Chapter 3

Some constructions in algebra

In this chapter more notions and notations from commutative algebra are first reviewed. Then polynomial system solving over finite fields is briefly discussed, with other background knowledge related to finite fields.

3.1 Commutative algebra

3.1.1 Ideal arithmetic

Definition 3.1.1. For two ideals $\mathfrak{a}, \mathfrak{b} \subset \mathbb{K}[\boldsymbol{x}]$, the set

$$\mathfrak{a} + \mathfrak{b} = \{F + G : \ F \in \mathfrak{a}, G \in \mathfrak{b}\}$$

is call the *sum* of $\mathfrak{a}$ and $\mathfrak{b}$, denoted by $\mathfrak{a} + \mathfrak{b}$.

This set can be shown to be an ideal. In particular, $\mathfrak{a}+\mathfrak{b}$ is the smallest ideal that contains both $\mathfrak{a}$ and $\mathfrak{b}$. In terms of corresponding varieties, $\mathsf{V}(\mathfrak{a} + \mathfrak{b}) = \mathsf{V}(\mathfrak{a}) \cap \mathsf{V}(\mathfrak{b})$. For $\mathfrak{a} = \langle F_1, \ldots, F_r \rangle$ and $\mathfrak{b} = \langle G_1, \ldots, G_s \rangle$, it can also be shown that $\mathfrak{a} + \mathfrak{b} = \langle F_1, \ldots, F_r, G_1, \ldots, G_s \rangle$.

Definition 3.1.2. For two ideals $\mathfrak{a}, \mathfrak{b} \subset \mathbb{K}[\boldsymbol{x}]$, their *product* is the ideal generated by $F \cdot G$ for all $F \in \mathfrak{a}$ and $G \in \mathfrak{b}$. We denote the product of $\mathfrak{a}$ and $\mathfrak{b}$ by $\mathfrak{a} \cdot \mathfrak{b}$.

From the definition, one can easily show that for $\mathfrak{a} = \langle F_1, \ldots, F_r \rangle$ and $\mathfrak{b} = \langle G_1, \ldots, G_s \rangle$, $\mathfrak{a} \cdot \mathfrak{b} = \langle F_i G_j : \ i = 1, \ldots, r, j = 1, \ldots, s \rangle$. As regards to their varieties, $\mathsf{V}(\mathfrak{a} \cdot \mathfrak{b}) = \mathsf{V}(\mathfrak{a}) \cup \mathsf{V}(\mathfrak{b})$.

Definition 3.1.3. For two $\mathfrak{a}, \mathfrak{b} \subset \mathbb{K}[\boldsymbol{x}]$, the set

$$\{F \in \mathbb{K}[\boldsymbol{x}] : FG \in \mathfrak{a}, \forall G \in \mathfrak{b}\}$$

is called the *ideal quotient* of $\mathfrak{a}$ by $\mathfrak{b}$ and denoted by $\mathfrak{a} : \mathfrak{b}$.

One can prove the set $\mathfrak{a} : \mathfrak{b}$ to be an ideal. In particular, $\mathfrak{a} \subset \mathfrak{a} : \mathfrak{b}$. To study the relationship on varieties of the three ideals $\mathfrak{a}$, $\mathfrak{b}$ and $\mathfrak{a} : \mathfrak{b}$, we need to introduce another important concept about varieties, the Zariski Closure.

Definition 3.1.4. Let $S \subset \mathbb{K}^n$. Then the *Zariski Closure* of S is the smallest affine variety that contains S, and it is denoted by $\overline{S}$.

One can show that $\overline{S} = \mathsf{V}(\mathsf{I}(S))$. Then the following proposition states the relationship on varieties about ideal quotients.

Proposition 3.1.1. *For the ideals $\mathfrak{a}, \mathfrak{b} \in \mathbb{K}[\boldsymbol{x}]$, we have $\mathsf{V}(\mathfrak{a} : \mathfrak{b}) \supset \overline{\mathsf{V}(\mathfrak{a}) - \mathsf{V}(\mathfrak{b})}$.*

Definition 3.1.5. The intersection $\mathfrak{a} \cap \mathfrak{b}$ of two ideals $\mathfrak{a}, \mathfrak{b} \subset \mathbb{K}[\boldsymbol{x}]$ in the sense of sets is also an ideal, and is called the *intersection* of $\mathfrak{a}$ and $\mathfrak{b}$.

Proposition 3.1.2. *For two ideals $\mathfrak{a}, \mathfrak{b} \subset \mathbb{K}[\boldsymbol{x}]$, we have*

$$\mathfrak{a} \cap \mathfrak{b} = (t\mathfrak{a} + (1 - t)\mathfrak{b}) \cap \mathbb{K}[\boldsymbol{x}],$$

where t is a newly introduced variable.

This proposition furnishes a computational approach to construct the intersection of two ideals by using Gröbner bases (with the Elimination Theorem).

3.1.2 Radical ideals and ideal-variety correspondence

The kind of ideals which corresponds to varieties are indeed radical ones. Now we exploit such ideals and their correspondence to the varieties.

Definition 3.1.6. An ideal $\mathfrak{a}$ is said to be *radical* if any F such that $F^m \in \mathfrak{a}$ for some integer $m \geq 1$ also satisfies $F \in \mathfrak{a}$.

In fact one can prove that for a variety $V \subset \mathbb{K}^n$, if $F^m \in \mathsf{I}(V)$, then $F \in \mathsf{I}(V)$. By the definition of radical ideals, clearly $\mathsf{I}(V)$ is radical. For an ideal which is not radical, we can perform the operation of taking its radical in the sense as follows.

Definition 3.1.7. The *radical* of an ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ is the set

$$\{F : F^m \in \mathfrak{a} \text{ for some integer } m \geq 1\}.$$

The radical of $\mathfrak{a}$ is denoted by $\sqrt{\mathfrak{a}}$.

The radical of an ideal is also an ideal. Now we can state the Nullstellensatz which relates varieties to radical ideals.

Theorem 3.1.3 (Hilbert's Strong Nullstellensatz). *Let $\mathbb{K}$ be an algebraically closed field. Then for any ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$, $\mathsf{I}(\mathsf{V}(\mathfrak{a})) = \sqrt{\mathfrak{a}}$.*

Theorem 3.1.4. *Let $\mathbb{K}$ be an algebraically closed field. Then the map I and V establishes an inclusion-reversing bijections between the set of affine varieties in $\mathbb{K}^n$ and that of radical ideals in $\mathbb{K}[\boldsymbol{x}]$.*

By inclusion-reversing we mean:

(a) If $V_1 \subset V_2$ for two varieties V_1 and V_2, we have $\mathsf{I}(V_1) \supset \mathsf{I}(V_2)$;

(b) If $\mathfrak{a}_1 \subset \mathfrak{a}_2$ for two radical ideals $\mathfrak{a}_1$ and $\mathfrak{a}_2$, we have $\mathsf{V}(\mathfrak{a}_1) \supset \mathsf{V}(\mathfrak{a}_2)$.

This one-one correspondence between varieties and radical ideals entitles us an important bridge to work on the other in order to study one of the two. For example, given a radical ideal, we can translate to its corresponding variety to have a geometric intuition about this ideal. On the other hand, the translation to ideals which are much easier to manipulate because of the algebraic form makes possible the computational study on varieties.

For a principal ideal, taking its radical is equivalent to extracting the squarefree part of its generator.

Proposition 3.1.5. *For $F \in \mathbb{K}[x]$, $\sqrt{\langle F \rangle} = \langle \tilde{F} \rangle$, where $\tilde{F}$ is the squarefree part of F.*

The following proposition reduces the problem of testing whether an element is in a radical ideal to that of testing an ideal membership, which can be solved via Gröbner bases.

Proposition 3.1.6. *Let* $\mathfrak{a} = [F_1, \ldots, F_r] \subset \mathbb{K}[\boldsymbol{x}]$ *be an ideal, and* F *a polynomial in* $\mathbb{K}[\boldsymbol{x}]$*. Then* $F \in \sqrt{\mathfrak{a}}$ *if and only if* $1 \in \langle F_1, \ldots, F_r, 1 - yF \rangle \subset \mathbb{K}[x_1, \ldots, x_n, y]$*.*

3.1.3 Prime decomposition

Similar to the idea behind factorization of polynomials, we also want to decompose ideals or varieties into smaller ones which can not be further decomposed. Next we first make clear those ideals and varieties which can not be reduced, and then relate ideals or varieties to their decomposition into such special ideals and varieties.

Definition 3.1.8. An ideal $\mathfrak{p} \subset \mathbb{K}[\boldsymbol{x}]$ is said to be *prime* if for any $F, G \in \mathbb{K}[\boldsymbol{x}]$, $FG \in \mathfrak{p}$ implies $F \in \mathfrak{p}$ or $G \in \mathfrak{p}$.

This property of prime ideals is similar to that of prime elements in a commutative ring with unity. Furthermore, from the definition, one can prove that a prime ideal is radical.

Definition 3.1.9. A variety $V \subset \mathbb{K}^n$ is called *irreducible* if V cannot be written as the union of two proper sub varieties, i.e., $V = V_1 \cup V_2$ implies $V = V_1$ or $V = V_2$.

As one can see, the definitions of prime ideals and irreducible varieties are similar. In fact, a one-one correspondence exists between them.

Proposition 3.1.7. *A variety* $V \subset \mathbb{K}^n$ *is irreducible if and only if* $\mathsf{I}(V)$ *is a prime ideal in* $\mathbb{K}[\boldsymbol{x}]$*.*

Proposition 3.1.8. *Let* $\mathbb{K}$ *be an algebraically closed field. Then the maps* I *and* V *induce a one-one correspondence between irreducible varieties in* $\mathbb{K}^n$ *and prime ideals in* $\mathbb{K}[\boldsymbol{x}]$*.*

A proper ideal in $\mathbb{K}[\boldsymbol{x}]$ is called *maximal* if there is no proper ideal containing it. A maximal ideal is prime. The following proposition characterizes maximal ideals.

Proposition 3.1.9. *The following statements hold:*

(a) Let $\mathbb{K}$ be a field. Then the ideal $\langle x_1 - a_1, \ldots, x_n - a_n \rangle$ in $\mathbb{K}[\boldsymbol{x}]$ with $a_i \in \mathbb{K}$ $(i = 1, \ldots, n)$ is maximal.

(b) If $\mathbb{K}$ is algebraically closed, then any maximal ideals in $\mathbb{K}[\boldsymbol{x}]$ is of the form $\langle x_1 - a_1, \ldots, x_n - a_n \rangle$ for some $a_i \in \mathbb{K}$ $(i = 1, \ldots, n)$.

Corollary 3.1.10. *Let $\mathbb{K}$ be an algebraically closed field. There exists a one-one correspondence between points in $\mathbb{K}^n$ and maximal ideals in $\mathbb{K}[\boldsymbol{x}]$.*

The next theorem states the possibility to decompose any radical ideals into the intersection of prime ideals.

Theorem 3.1.11. *Every radical ideal $\mathfrak{a} \in \mathbb{K}[\boldsymbol{x}]$ can be written as the intersection of finite prime ideals $\mathfrak{p}_1, \ldots, \mathfrak{p}_r$, i.e., $\mathfrak{a} = \bigcap_{i=1}^r \mathfrak{p}_i$. Furthermore, if we require $\mathfrak{p}_i \not\subset \mathfrak{p}_j$ for $i \neq j$, then the ideals $\mathfrak{p}_1, \ldots, \mathfrak{p}_r$ are unique up to their orders.*

Translating into the variety language with the ideal-variety correspondence, we have the following corollary.

Corollary 3.1.12. *Every variety $V \subset \mathbb{K}^n$ can be written as $V = \bigcup_{i=1}^r V_i$ with V_i irreducible variety in $\mathbb{K}^n$ for $i = 1, \ldots, n$. Furthermore, if we require $V_i \not\subset V_j$ for $i \neq j$, the varieties $V_1, \ldots, V_r$ are unique up to their orders.*

The decomposition $\mathfrak{a} = \bigcap_{i=1}^r \mathfrak{p}_i$ and $V = \bigcup_{i=1}^r V_i$ as in Theorem 3.1.11 and Corollary 3.1.12 are called a *minimal decomposition* of $\mathfrak{a}$ and V. Now we try the special case when the radical ideal is generated by a squarefree polynomial $F \in \mathbb{K}[\boldsymbol{x}]$. Let $F = F_1, \ldots, F_r$ be a factorization of F. Then $\langle F \rangle = \bigcap_{i=1}^r \langle F_i \rangle$ is a minimal decomposition of $\mathfrak{a}$, for each F_i is irreducible over $\mathbb{K}$ and thus $\langle F_i \rangle$ is prime.

We end this section with the following Chinese Remainder Theorem (short as CRT), which will be frequently used.

Theorem 3.1.13 (Chinese Remainder Theorem). *Let $\mathfrak{a}_1, \ldots, \mathfrak{a}_r$ be ideals in $\mathcal{R}$ such that $\mathfrak{a}_i$ and $\mathfrak{a}_j$ are coprime (namely $\mathfrak{a}_i + \mathfrak{a}_j = \mathcal{R}$) for all $i \neq j$. Then*

$$\mathcal{R}/\Big(\bigcap_{i=1}^r \mathfrak{a}_i\Big) \cong \prod_{i=1}^r \mathcal{R}/\mathfrak{a}_i.$$

3.2 Basics of finite fields

The study on finite fields can be dated back to the 17th and 18th century, but finite fields only become of great interest in the 20th century because of its wide applications in cryptography, Coding Theory, combinatorics and etc. In this section we first provide basic results on the structure of finite fields, and then present notions and algorithms related to the scope of the thesis.

A field of finite elements is called a *finite field*. The number of elements a finite field contains can be characterized by the following theorem.

Theorem 3.2.1 ([101, Theorem 2.2]). *Let $\mathbb{K}$ be a finite field. Then $\mathbb{K}$ has p^n elements, where the prime p is the characteristic of $\mathbb{K}$ and n is the degree of $\mathbb{K}$ over its prime subfield.*

Proposition 3.2.2. *If $\mathbb{K}$ is a finite field with q elements, then every $a \in \mathbb{K}$ satisfies $a^q = a$.*

For every prime p and positive integer n there exists a finite field with p^n elements. It can also be concluded that two finite fields are isomorphic if and only if their numbers of elements are equal. Therefore there only exists one finite field of $q = p^m$ elements up to isomorphisms, and we denote it by $\mathbb{F}_q$.

For a field $\mathbb{K}$, all the elements $\mathbb{K} \setminus \{0\}$ form a group w.r.t. the multiplications in $\mathbb{K}$, and this group is called the *multiplicative group* of $\mathbb{K}$, denoted by $\mathbb{K}^*$. For a finite field $\mathbb{F}_q$, it can be proved that $\mathbb{F}_q^*$ is cyclic.

3.2.1 Polynomial system solving over finite fields

In general, solving a polynomial equation set $\mathcal{F} = 0$ for $\mathcal{F} \subset \mathbb{K}[\boldsymbol{x}]$ means to find all its solutions in $\tilde{\mathbb{K}}^n$, where $\tilde{\mathbb{K}}$ is an extension of $\mathbb{K}$. Usually the solutions in the algebraic closure of $\mathbb{K}$ are of interest. However, for polynomial system solving over a finite field $\mathbb{F}_q$, compared with the solutions in $\overline{\mathbb{F}}_q^n$, those in $\mathbb{F}_q^n$ are usually more important because of their practical background in applications like cryptography and Coding Theory.

It is obvious that the number of solutions of $\mathcal{F} = 0$ in $\mathbb{F}_q^n$ is finite, and thus one trivial method to compute them is traversing all possibilities to check whether they

satisfy $\mathcal{F} = 0$. This method is useful when both the cardinality q and the number of variable n are small. However, when the number of possible solutions is large (for example, in multivariate public-key cryptography the solution space is usually of cardinality greater than 2^{160}), the traversal method is no longer applicable. In this case, one has to turn to other solving methods like those based on Gröbner bases and triangular sets.

A common strategy to restrict the solutions of $\mathcal{F} = 0$ in $\mathbb{F}_q^n$ is to solve $\mathcal{F} \cup \mathcal{P} = 0$ instead, where $\mathcal{P} = \{x_1^q - x_1, \ldots, x_n^q - x_n\}$ is the set of field polynomials of $\mathbb{F}_q$. It is easy to prove that this strategy works as intended. To facilitate polynomial system solving over $\mathbb{F}_2$, the Computer Algebra system Magma introduces the Boolean polynomial ring defined over $\mathbb{F}_2$ whose polynomials are automatically reduced with the field relations $x_i^2 = x_i$ for $i = 1, \ldots, n$.

Most methods based on Gröbner bases and triangular sets for solving polynomial systems over fields of characteristic 0 are also applicable to systems over finite fields with the above strategy (Clearly RUR does not work any longer). However, specialized algorithms for computing Gröbner bases (e.g., the XL algorithm) and triangular sets over finite fields have also been designed taking the structure of finite fields into consideration [32, 3, 64, 15], and these algorithms, which are usually more efficient, should be chosen when applied to solve polynomial systems over finite fields. Furthermore, algorithms for solving SAT problems can also be used to solve polynomial systems over $\mathbb{F}_2$ [40, 117]. In particular SAT solvers are effective tools to achieve this purpose [47, 142].

3.2.2 Squarefree decomposition over finite fields

Squarefree decomposition is a fundamental tool to understand the inherent structure of a polynomial. For a principal ideal domain, there is a close relationship between radical ideals and squarefree polynomials: an ideal is radical if and only if its generating polynomial is squarefree. We will use this result later, but over a specific ring. Furthermore, squarefree decomposition is usually a proceeding step for polynomial factorization, and thus in factorization we usually assume the input polynomial is squarefree.

Here we recall the method of computing squarefree decompositions of univariate

polynomials over finite fields from [68], which is the starting point for our discussions in Chapters 5 and 6.

For a non-constant polynomial $F \in \mathbb{K}[x]$, we call $\{[A_1, a_1], \ldots, [A_s, a_s]\}$ a *squarefree decomposition* of F, where $A_i \in \mathbb{K}[\boldsymbol{x}]$ is non-constant and a_i is a positive integer for $i = 1, \ldots, s$, if the following conditions hold:

- $F = cA_1^{a_1} \cdots A_s^{a_s}$ for some nonzero constant $c \in \mathbb{K}$;
- A_i is squarefree for all $i = 1, \ldots, s$;
- $\gcd(A_i, A_j) = 1$ for all $i \neq j$.

In particular, the polynomial $A_1 \cdots A_s$ is the squarefree part of F.

Algorithms for squarefree decomposition of polynomials over finite fields are somehow different from those over fields of characteristic 0 in the sense that the derivative, an important tool in squarefree decomposition, of any pth power over a field of characteristic p is 0.

Proposition 3.2.3. *Let $\mathbb{K}$ be a field of characteristic $p > 0$. For any $F \in \mathbb{K}[x]$, there exist unique (up to unit) polynomials $P_1, \ldots, P_k$ and Q in $\mathbb{K}[x]$ such that*

(a) $F = Q \prod_{i=1}^{k} P_i^i$;

(b) $\gcd(P_i, P_i') = 1$, *which means that P_i is squarefree for all $i = 1, \ldots, k$;*

(c) $\gcd(P_i, P_j) = \gcd(P_i, Q) = 1$ *for all $i \neq j$;*

(d) $Q' = 0$;

(e) if $i \equiv 0 \mod p$, *then* $P_i = 1$.

Corollary 3.2.4. *Let $\mathbb{K}$ be a field of characteristic $p > 0$ and $F = Q \prod_{i=1}^{k} P_i^i$ be the decomposition given in Proposition 3.2.3. Then* $\gcd(F, F') = Q \prod_{i=1}^{k} P_i^{i-1}$.

Proposition 3.2.5. *Let $\mathbb{K}$ be a field of characteristic $p > 0$ and $F \in \mathbb{K}[x]$. Then $F' = 0$ if and only if there exists a polynomial $G \in \mathbb{K}[x]$ such that $F(x) = G(x^p)$.*

For any $F \in \mathbb{F}_q[x] \setminus \mathbb{F}_q$, let $F = Q\prod_{i=1}^{k} P_i^i$ be the decomposition in Proposition 3.2.3. When $Q \in \mathbb{F}_q[x] \setminus \mathbb{F}_q$, the squarefree part of F is $Q_1 \prod_{i=1}^{k} P_i$, where Q_1 is the squarefree part of Q. However, by Corollary 3.2.4,

$$\gcd(F, F') = Q\prod_{i=1}^{k} P_i^{i-1} \quad \text{and} \quad F/\gcd(F, F') = \prod_{i=1}^{k} P_i.$$

Hence using $F/\gcd(F, F')$ to obtain the squarefree part will cause Q_1 missing. This problem can be solved by the following squarefree decomposition algorithm (Algorithm 8). In this and other algorithms, the operation

$$\mathsf{merge}(\{[A_1, a_1], \ldots, [A_s, a_s]\}, \{[D_1, d_1], \ldots, [D_t, d_t]\})$$

first merges the two sets into one and then replaces any $[A_i, a_i]$ and $[D_j, d_j]$ for which $A_i = D_j$ by $[A_i, a_i + d_j]$.

Algorithm 8: Squarefree decomposition of a univariate polynomial over a finite field $\mathbb{S} := \mathsf{sqf}(F)$

Input: F — a polynomial in $\mathbb{F}_q[x] \setminus \mathbb{F}_q$.

Output: $\mathbb{S}$ — the squarefree decomposition of F.

8.1 $\mathbb{S} := \emptyset$; $d := 1$;
8.2 $C_1 := \gcd(F, F')$;
8.3 $B_1 := F/C_1$;
8.4 **while** $B_1 \in \mathbb{F}_q[x] \setminus \mathbb{F}_q$ **do**
8.5 $\quad B_2 := \gcd(B_1, C_1)$, $C_2 := C_1/B_2$, $P := B_1/B_2$;
8.6 $\quad$ **if** $P \in \mathbb{F}_q[x] \setminus \mathbb{F}_q$ **then** $\mathbb{S} := \mathbb{S} \cup \{[P, d]\}$;
8.7 $\quad B_1 := B_2$; $C_1 := C_2$;
8.8 $\quad d := d + 1$;
8.9 **end**
8.10 **if** $C_1 \in \mathbb{F}_q[x] \setminus \mathbb{F}_q$ **then**
8.11 $\quad C_3 :=$ the pth root of C_1;
8.12 $\quad \{[A_1, a_1], \ldots, [A_s, a_s]\} := \mathsf{sqf}(C_3)$;
8.13 $\quad \mathbb{S} := \mathsf{merge}(\{[A_1, a_1 \cdot p], \ldots, [A_s, a_s \cdot p]\}, \mathbb{S})$;
8.14 **end**

3.2.3 Multiple derivations

Multiple derivations are a tool we will use for verifying pth powers in the extended squarefree decomposition we will discuss in Chapters 5 and 6.

Let $\mathcal{R}$ be a ring. A map $D : \mathcal{R} \to \mathcal{R}$ is called a *derivation* of $\mathcal{R}$ if for all $a, b \in \mathcal{R}$, $D(a+b) = D(a) + D(b)$ and $D(ab) = D(a)b + aD(b)$. The set of all derivations of $\mathcal{R}$, denoted by $\mathrm{Der}(\mathcal{R})$, is an $\mathcal{R}$-module. One can also show that for any ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$, a derivation $D \in \mathrm{Der}(\mathbb{K}[\boldsymbol{x}])$ induces a derivation in $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{a})$ if $D(\mathfrak{a}) \subset \mathfrak{a}$.

If a basis of the $\mathbb{K}[\boldsymbol{x}]$-module $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}])$ is known, then the proposition below furnishes an approach for computing a generating basis of $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{a})$ with syzygy computation.

Proposition 3.2.6 ([62, Proposition 5.2]). *Let $D_1, \ldots, D_r$ be a basis of $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}])$, $\mathfrak{a} = \langle F_1, \ldots, F_s \rangle \subset \mathbb{K}[\boldsymbol{x}]$, and $v = (v_1, \ldots, v_r) \in \mathbb{K}[\boldsymbol{x}]^r$. Then a derivation $D = \sum_{i=1}^{r} v_i D_i$ satisfies $D(\mathfrak{a}) \subset \mathfrak{a}$ if and only if $D(F_j) = \sum_{i=1}^{r} v_i D_i(F_j) = 0 \mod \mathfrak{a}$ for $j = 1, \ldots, s$.*

In Chapters 5 and 6 we will often work on $\tilde{\mathbb{K}}[\boldsymbol{y}] := \mathbb{K}(u_1, \ldots, u_t)[y_1, \ldots, y_r]$, where $u_1, \ldots, u_t$ and $y_1, \ldots, y_r$ are transcendental and algebraic elements over $\mathbb{K}$ defined by a regular set $\mathcal{T}$ respectively. In this case $\mathrm{Der}(\tilde{\mathbb{K}}[\boldsymbol{y}]/\operatorname{sat}(\mathcal{T})_{\tilde{\mathbb{K}}})$ is of our interest and a basis of $\mathrm{Der}(\tilde{\mathbb{K}})$ is $D_i := \partial/\partial u_i$, for $i = 1, \ldots, t$ [68]. Then a basis $\tilde{D}_1, \ldots, \tilde{D}_{t+r}$ of $\mathrm{Der}(\tilde{\mathbb{K}}[\boldsymbol{y}])$ can be obtained in the following way [62]: for any $P = \sum_{\boldsymbol{i} \in \mathbb{N}^r} c_{\boldsymbol{i}} \boldsymbol{y}^{\boldsymbol{i}}$, define

$$\tilde{D}_i(P) = \frac{\partial P}{\partial y_i}, \quad i = 1, \ldots, r;$$
$$\tilde{D}_i(P) = \sum_{\boldsymbol{i} \in \mathbb{N}^r} D_{i-r}(c_{\boldsymbol{i}}) \boldsymbol{y}^{\boldsymbol{i}}, \quad i = r+1, \ldots, r+t.$$

Therefore, from any regular set $\mathcal{T} \subset \mathbb{K}[\boldsymbol{x}]$, one can construct a basis of $\mathrm{Der}(\tilde{\mathbb{K}}[\boldsymbol{y}]/\operatorname{sat}(\mathcal{T})_{\tilde{\mathbb{K}}})$.

The following results show how to detect pth powers in a finitely generated field extension of a finite field and a quotient ring with multiple derivations.

Proposition 3.2.7 ([86, VIII, Proposition 5.4]). *Let $\mathbb{K}$ be a finitely generated extension of a perfect field. Then $P \in \mathbb{K}$ is a pth power in $\mathbb{K}$ if and only if $D(P) = 0$ for all $D \in \mathrm{Der}(\mathbb{K}[\boldsymbol{x}])$.*

Corollary 3.2.8 ([62, Proposition 5.3]). *Let $\mathbb{K}$ be a field of characteristic $p > 0$, $\mathfrak{a}$ a zero-dimensional radical ideal in $\mathbb{K}[\boldsymbol{x}]$, and $P \in \mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$. Then there exists a $Q \in \mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$ such that $Q^p = P$ if and only if $D(P) = 0$ for all $D \in \mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{a})$.*

With multiple derivations Proposition 3.2.3 which is useful for squarefree decomposition over finite fields may be generalized to fields of positive characteristics. The following proposition will be used in Section 6.2.2.

Proposition 3.2.9 ([68, Proposition 8]). *Let $\mathbb{K}$ be a field of characteristic $p > 0$, and $D_1, \ldots, D_t$ a basis of $\mathrm{Der}(\mathbb{K}[z])$. Then for a polynomial $F \in \mathbb{K}[z]$, there exist polynomial P_i and $Q \in \mathbb{K}[z]$ such that*

(a) $F = Q \prod_{i=1}^{k} P_i^i$ *with P_i squarefree;*

(b) $\gcd(P_i, P_j) = \gcd(Q, P_i) = 1$ *for $i \neq j$;*

(c) $D_i(Q) = 0$ *for $i = 1, \ldots, t$;*

(d) if p divides i, then $P_i = 1$;

(e) $\gcd(f, D_1(f), \ldots, D_t(f)) = Q \prod_i P_i^{i-1}$.

3.2.4 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 [138, 139, 137]. 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 [36, 72]. Combining with the Feng–Rao majority voting algorithm [59], this algorithm can often decode codes with more with $(d_{\min} - 1)/2$ errors if the error locations are general [17], 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 $\boldsymbol{u} = (u_1, \ldots, u_n) \in \mathbb{Z}_{\geq 0}^n$ and a term $\boldsymbol{x}^{\boldsymbol{u}} = x_1^{u_1} \cdots x_n^{u_n} \in \mathbb{K}[\boldsymbol{x}]$ are 1–1 corresponding, usually we do not distinguish one from the other. A mapping $\mathrm{E} : \mathbb{Z}_{\geq 0}^n \longrightarrow \mathbb{K}$ is called an *n-dimensional array*. In Coding Theory, the array E is usually a syndrome array determined by the error word [137]. Besides the term ordering, we define the following partial ordering: for two terms $\boldsymbol{u} = (u_1, \ldots, u_n)$ and $\boldsymbol{v} = (v_1, \ldots, v_n)$, we say that $\boldsymbol{u} \prec \boldsymbol{v}$ if $u_i \leq v_i$ for $i = 1, \ldots, n$.

Definition 3.2.1. Given a polynomial $F = \sum_{\boldsymbol{s}} F_{\boldsymbol{s}} \boldsymbol{x}^{\boldsymbol{s}} \in \mathbb{K}[\boldsymbol{x}]$, an n-dimensional mapping E is said to satisfy the *n-dimensional linearly recurring relation* with *characteristic polynomial* F if

$$\sum_{\boldsymbol{s}} F_{\boldsymbol{s}} \mathrm{E}_{\boldsymbol{s}+\boldsymbol{r}} = 0, \quad \forall \boldsymbol{r} \succ 0. \tag{3.1}$$

The set of all characteristic polynomials of the n-dimensional linearly recurring relation for the array E forms an ideal, denoted by $\mathsf{I}(\mathrm{E})$. 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 $\mathsf{I}(\mathrm{E})$ used here in this thesis follows [137], and one can easily see that this definition is equivalent to that in [36] by [137, Thereom 23].

Furthermore, the set of minimal polynomials for $\mathsf{I}(\mathrm{E})$ w.r.t. $<$, which the BMS algorithm computes, is actually the Gröbner basis of $\mathsf{I}(\mathrm{E})$ w.r.t. $<$ [139, Lemma 5]. The canonical basis of $\mathbb{K}[\boldsymbol{x}]/\mathsf{I}(\mathrm{E})$ is also called the *delta set* of E, denoted by $\Delta(\mathrm{E})$. The term "delta set" comes from the property that if $\boldsymbol{u} \in \mathbb{Z}_{\geq 0}^n$ is contained in $\Delta(\mathrm{E})$, then $\Delta(\mathrm{E})$ also contains all elements $\boldsymbol{v} \in \mathbb{Z}_{\geq 0}^n$ 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 $\mathrm{lt}(F) = \boldsymbol{s}$ is said to be *valid for* E *up to* $\boldsymbol{u}$ if either $\boldsymbol{u} \not\succ \boldsymbol{s}$ or

$$\sum_{\boldsymbol{t}} F_{\boldsymbol{t}} \mathrm{E}_{\boldsymbol{t}+\boldsymbol{r}} = 0, \quad \forall \boldsymbol{r} \ (0 \prec \boldsymbol{r} \leq \boldsymbol{u} - \boldsymbol{s}).$$

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

Similarly to FGLM, the BMS algorithm also handles terms in $\mathbb{K}[\boldsymbol{x}]$ one by one according to $<$, so that the polynomial set $\mathcal{F}$ it maintains is valid up to the new term. Suppose $\mathcal{F}$ is valid up to some term $\boldsymbol{u}$. When the next term of $\boldsymbol{u}$ w.r.t.

$<$, denoted by Next($\boldsymbol{u}$), is considered, the BMS algorithm will update $\mathcal{F}$ so that it keeps valid up to Next($\boldsymbol{u}$). Meanwhile, terms determined by Next($\boldsymbol{u}$) are also tested whether they are members of Δ(E). Therefore, more and more terms will be verified in Δ(E) as the BMS algorithm proceeds. The set of verified terms in Δ(E) after the term $\boldsymbol{u}$ is called the *delta set up to* $\boldsymbol{u}$ and denoted by $\Delta(\boldsymbol{u})$. Then we have

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

After a certain number of terms are considered, $\mathcal{F}$ and $\Delta(\boldsymbol{u})$ will grow to the Gröbner basis of I(E) and Δ(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 9 for convenience of later use. More details will also be provided in Section 4.2.2. One may refer to [137, 36] for a detailed description. In Algorithm 9 below, the polynomial set $\mathcal{G}$, called the *witness set*, is auxiliary and will not be returned with $\mathcal{F}$ in the end of the BMS algorithm.

Algorithm 9: $(F^+, G^+) := \text{BMSUpdate}(F, G, \text{Next}(\boldsymbol{u}), \text{E})$

Input:

$\mathcal{F}$, a minimal polynomial set valid up to $\boldsymbol{u}$;

$\mathcal{G}$, a witness set up to $\boldsymbol{u}$;

Next($\boldsymbol{u}$), a term;

E, a n-dimensional array up to Next($\boldsymbol{u}$).

Output:

$\mathcal{F}^+$, a minimal polynomial set valid up to Next($\boldsymbol{u}$);

$\mathcal{G}^+$, a witness set up to Next($\boldsymbol{u}$).

(a) Test whether every polynomial in $\mathcal{F}$ is valid up to Next($\boldsymbol{u}$)

(b) Update $\mathcal{G}^+$ and compute the new delta set up to Next($\boldsymbol{u}$) accordingly

(c) Construct new polynomials in $\mathcal{F}^+$ such that they are valid up to Next($\boldsymbol{u}$)

Part II

Contributions

Chapter 4

Sparse FGLM algorithms

The term ordering plays an important role in the theory of Gröbner bases. It has been shown in previous contents of the thesis that the change of orderings of Gröbner bases from DRL to LEX is one important step in the Gröbner basis method for solving polynomial systems. Furthermore, some practical problems can be directly modeled as the change of orderings of Gröbner bases, for example [22, 104] from cryptography and Coding Theory.

However, the computation of Gröbner bases greatly enhanced recently [49, 50], the step to change the orderings of Gröbner bases has become the bottleneck of the whole solving process (see Section 4.4 for details). Hence it is of crucial significance to design efficient algorithms for the change of ordering. The purpose of this chapter is precisely to provide such efficient algorithms.

Suppose $\mathcal{G}_1$ is the Gröbner basis of a zero-dimensional ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ w.r.t. a term ordering $<_1$. Given another term ordering $<_2$, one wants to compute the Gröbner basis $\mathcal{G}_2$ of $\mathfrak{a}$ w.r.t. it. Denote by D the degree of $\mathfrak{a}$, that is, the dimension of $\mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$ as a vector space. These notations are fixed hereafter in this chapter.

This chapter is organized as follows. Section 4.1 is devoted to the shape position case, where two methods with their complexity analyses are exploited. The method based on the BMS algorithm for the general case is presented in Section 4.2, with a top-level algorithm which combines all the previous methods. The sparsity of T_1 is studied in Section 4.3 and experimental results are provided in Section 4.4.

The results in this chapter are based on the joint work with Jean-Charles Faugère. Part of the contents has been published in [55].

4.1 Ideals in shape position

Definition 4.1.1. An ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ is said to be *in shape position* if its Gröbner basis w.r.t LEX is of the following form

$$[F_1(x_1), x_2 - F_2(x_1), \ldots, x_n - F_n(x_1)]. \tag{4.1}$$

One may easily see that $\mathfrak{a}$ here is zero-dimensional and $\deg(F_1) = D$.

Such ideals take a large proportion in all the consistent ideals and have been well studied and applied [13, 135]. 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 the multiplication matrix T_1 is nonsingular. Otherwise, one knows by the Stichelberger's theorem (see, e.g. [135, Theorem 2.1]) that $x_1 = 0$ will be a root of the univariate polynomial in $\mathfrak{a}$'s Gröbner basis w.r.t. LEX, and sometimes the polynomial system can be further simplified.

4.1.1 Probabilistic algorithm to compute Gröbner basis of the ideal

Algorithm description

Given a zero-dimensional ideal $\mathfrak{a}$, if the univariate polynomial $F_1(x_1)$ in its Gröbner basis w.r.t. LEX is of degree D, then we know $\mathfrak{a}$ is in shape position.

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

$$s = [\langle \boldsymbol{r}, T_1^i \boldsymbol{e} \rangle : i = 0, \ldots, 2D-1], \tag{4.2}$$

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 $\mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$, 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 $\mathfrak{a}$ can be verified to be in shape position.

Suppose $\deg(\tilde{F}_1) = D$ holds and F_i in (4.1) is of the form $F_i = \sum_{k=0}^{D-1} c_{i,k} x_1^k$ for $i = 2, \ldots, n$. Then computing the whole Gröbner basis of $\mathfrak{a}$ 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 4.1.2. Let $s = [s_0, s_1, s_2, \cdots]$ be a sequence of elements in $\mathbb{K}$ and d an integer. 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 4.1.1 ([76]). *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) $d = \operatorname{rank}(H_d(s)) = \operatorname{rank}(H_i(s))$ *for all* $i > d$;

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

For each $i = 2, \ldots, n$, as $x_i - \sum_{k=0}^{D-1} c_{i,k} x_1^k \in \mathfrak{a}$, one has $\operatorname{nform}(x_i - \sum_{k=0}^{D-1} c_{i,k} x_1^k) = 0$, 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 $j = 1, \ldots, D-1$, one can further construct D linear equations

$$\langle \boldsymbol{r}, T_1^j \boldsymbol{v}_i \rangle = \sum_{k=0}^{D-1} c_{i,k} \cdot \langle \boldsymbol{r}, T_1^{k+j} \boldsymbol{e} \rangle, \quad j = 0, \ldots, D-1. \tag{4.3}$$

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 4.1.1. Furthermore, the linear equation set (4.3) with the Hankel matrix H can be efficiently solved [18].

All the solutions of these linear systems for $i = 2, \ldots, n$ will lead to the Gröbner basis we want to compute.

The method above is summarized in the following algorithm, 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 [162].

Algorithm 10: Shape position (probabilistic) $\mathcal{G}_2 := \text{ShapePro}(\mathcal{G}_1, <_1)$

Input: $\mathcal{G}_1$, Gröbner basis of a zero-dimensional ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ w.r.t. $<_1$

Output: $\mathcal{G}_2$, Gröbner basis of $\mathfrak{a}$ w.r.t. LEX if the polynomial returned by BerlekampMassey() is of degree D; Fail, otherwise.

10.1 Compute the canonical basis of $\mathbb{K}[\boldsymbol{x}]/\langle \mathcal{G}_1 \rangle$ and multiplication matrices $T_1, \ldots, T_n$;
10.2 $\boldsymbol{e} := (1, 0, \ldots, 0)^t \in \mathbb{K}^{(D\times 1)}$;
10.3 Choose $\boldsymbol{r}_0 = \boldsymbol{r} \in \mathbb{K}^{(D\times 1)}$ randomly;
10.4 **for** $i = 1, \ldots, 2D-1$ **do**
10.5 $\quad \boldsymbol{r}_i := (T_1^t)\boldsymbol{r}_{i-1}$;
10.6 **end**
10.7 Generate the sequence $s := [\langle \boldsymbol{r}_i, \boldsymbol{e} \rangle : i = 0, \ldots, 2D-1]$;
10.8 $F_1 := \text{BerlekampMassey}(s)$;
10.9 **if** $\deg(F_1) = D$ **then**
10.10 $\quad H := H_D(s)$ // *Construct the Hankel matrix*
10.11 $\quad$ **for** $i = 2, \ldots, n$ **do**
10.12 $\quad\quad \boldsymbol{b} := (\langle \boldsymbol{r}_j, T_i \boldsymbol{e} \rangle : j = 0, \ldots, D-1)^t$;
10.13 $\quad\quad$ Compute $\boldsymbol{c} = (c_1, \ldots, c_D)^t := H^{-1}\boldsymbol{b}$;
10.14 $\quad\quad F_i := \sum_{k=0}^{D-1} \boldsymbol{c}_{k+1} x_1^k$;
10.15 $\quad$ **end**
10.16 $\quad$ **return** $[F_1, x_2 - F_2, \ldots, x_n - F_n]$;
10.17 **else**
10.18 $\quad$ **return** Fail;
10.19 **end**

Remark 4.1.1. *As can be seen from the description of Algorithm 10, 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 $\mathfrak{a}$ *is indeed in shape position.*

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 $\tilde{f}_1$ [162].

Next we show how the linear system (4.3) 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 (4.2) and (4.3)

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

Therefore, when computing the sequence (4.2), we can record $(T_1^t)^i\boldsymbol{r}$ $(i = 0, \ldots, 2D-1)$ and use them for construction of the linear equation set (4.3).

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 (4.2), 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 $\boldsymbol{v}_i = T_i\boldsymbol{e} = \mathrm{nform}(x_i)$. Without loss of generality, we can assume that $\mathrm{nform}(x_i) = x_i$ (this is not true only if there is a linear equation $x_i + \cdots$ in the Gröbner basis $\mathcal{G}_1$, 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 $i = 2, \ldots, n$, 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 [18]. Summarizing the analyses above, we have the following complexity result for this method.

Theorem 4.1.2. *Assume that* T_1 *is constructed (note that* $T_2, \ldots, T_n$ *are not needed). If the minimal polynomial of* (4.2) *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.

Illustrative example

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

$$\mathcal{G}_1 = [x_2^2 + 9\,x_2 + 2\,x_1 + 6,\ x_1^2 + 2\,x_2 + 9,\ x_3 + 9],$$

we first compute the degree of $\mathfrak{a}$ as $D = 4$, the canonical basis $B = [1, x_1, x_2, x_1x_2]$, and the multiplication matrices T_1, T_2 and T_3.

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 $\tilde{F}_1 = x_1^4 + 8\,x_1 + 9$. From the equality $\deg(\tilde{F}_1) = D = 4$, we know now that the input ideal $\mathfrak{a}$ 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 $x_2 - F_2(x_1) \in \mathcal{G}_2$ 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 $\mathcal{G}_2$ as $x_2 + 6\,x_1^2 + 10$. The other polynomial $x_3 - F_3(x_1)$ can be similarly computed. In the end, we have the Gröbner basis of $\mathfrak{a}$ w.r.t. LEX

$$\mathcal{G}_2 = [x_1^4 + 8\,x_1 + 9,\ x_2 + 6\,x_1^2 + 10,\ x_3 + 9].$$

4.1.2 Deterministic algorithm to compute Gröbner basis of radical of the ideal

As already explained in Remarks 4.1.1, 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 fact, the deterministic Wiedemann algorithm can be applied to obtain the univariate polynomial F_1, then one knows for sure whether $\mathfrak{a}$ 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 $\mathfrak{a}$. 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 $\mathfrak{a}$ in a way that the output is the Gröbner basis of $\sqrt{\mathfrak{a}}$.

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

$$[\langle \boldsymbol{e}_1, T_1^j \boldsymbol{e} \rangle : j = 0, \ldots, 2D-1]. \tag{4.4}$$

Suppose $d_1 = \deg(F_{1,1})$, and $\boldsymbol{b}_1 = F_{1,1}(T_1)\boldsymbol{e}$. If $\boldsymbol{b}_1 = \boldsymbol{0}$, 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 $\leq D - d_1$ (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 ($\leq D$) rounds and one finds $F_1 = F_{1,1} \cdots F_{1,r}$.

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

$$[\langle \boldsymbol{w}, T_1^i \boldsymbol{e}\rangle : i = 0, \ldots, 2D-1], \tag{4.5}$$

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 $\mathfrak{a} + \langle \tilde{F}_1\rangle$ w.r.t. LEX. Since the ideal $\mathfrak{a}$ is in shape position, it is not hard to see that the ideal $\mathfrak{a} + \langle \tilde{F}_1\rangle$ 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 (4.3)

$$\langle \boldsymbol{w}, T_1^j T_i \boldsymbol{e}\rangle = \sum_{k=0}^{d-1} y_k \cdot \langle \boldsymbol{w}, T_1^{k+j} \boldsymbol{e}\rangle, \quad j = 0, \ldots, d-1, \tag{4.6}$$

where $y_0, \ldots, y_{d-1}$ are the unknowns. As the $d \times d$ Hankel matrix of (4.5) is invertible by Theorem 4.1.1, there is a unique solution $c_{i,0}, c_{i,1}, \ldots, c_{i,d-1}$ for (4.6). Next we will connect this solution and a polynomial in the Gröbner basis of $\mathfrak{a} + \langle \tilde{F}_1\rangle$, and the following lemma is useful to show this connection.

Lemma 4.1.3. *Suppose $\tilde{F}_1$ is the minimal polynomial of (4.5) for some $\boldsymbol{w} \in \mathbb{K}^{(D\times 1)}$, $\tilde{T}_1$ the multiplication matrix of x_1 of the ideal $\mathfrak{a}+\langle \tilde{F}_1\rangle$ 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 $\mathbb{K}[\boldsymbol{x}]/(\mathfrak{a} + \langle \tilde{F}_1\rangle)$. 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. Suppose $\tilde{F}_1 = x_1^d + \sum_{k=0}^{d-1} a_k x_1^k$. Then according to the FGLM criteria, for the ideal $\mathfrak{a} + \langle \tilde{F}_1\rangle$,

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

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 4.1.4. *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 (4.5). 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* (4.6), *is in the Gröbner basis of* $\mathfrak{a} + \langle \tilde{F}_1 \rangle$ *w.r.t. LEX.*

Proof. Let $\tilde{T}_1, \ldots, \tilde{T}_d$ be the multiplication matrices of the ideal $\mathfrak{a} + \langle \tilde{F}_1 \rangle$ w.r.t. $<_1$.

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 $\mathfrak{a} + \langle \tilde{F}_1 \rangle$ 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

$$\langle \tilde{\boldsymbol{w}}, \tilde{T}_1^j \tilde{T}_i \tilde{\boldsymbol{e}} \rangle = \sum_{k=0}^{d-1} y_k \cdot \langle \tilde{\boldsymbol{w}}, \tilde{T}_1^{k+j} \tilde{\boldsymbol{e}} \rangle, \quad j = 0, \ldots, d-1. \tag{4.7}$$

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 (4.7) is invertible, and that the two linear equation sets (4.6) and (4.7) share the same solution. In particular, we will prove (4.6) and (4.7) are the same themselves for some $\tilde{\boldsymbol{w}}$.

To prove that, we need to show the two Hankel matrices and the vectors in the left hands of (4.6) and (4.7) 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 $j = 0, \ldots, 2d-2$;

(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 $j = 0, \ldots, d-1$.

Next we will prove these two arguments respectively.

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

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

As the vectors $\tilde{\boldsymbol{e}}, \tilde{T}_1 \tilde{\boldsymbol{e}}, \ldots, \tilde{T}_1^{d-1} \tilde{\boldsymbol{e}}$ are linearly independent, the above linear equation set has a unique solution $\overline{\boldsymbol{w}}$ for the unknown $\tilde{\boldsymbol{w}}$. From Lemma 4.1.3, the vector sequence $[\tilde{\boldsymbol{e}}, \tilde{T}_1 \tilde{\boldsymbol{e}}, \tilde{T}_1^2 \tilde{\boldsymbol{e}}, \ldots]$ and the sequence (4.5) share the same minimal polynomial $\tilde{F}_1$ of degree d. Thus there exist $a_0, \ldots, a_{d-1} \in \mathbb{K}$ 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 $j = d+1, \ldots, 2d-2$ 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 $j = 0, 1, \ldots$.

(ii) Since there is a polynomial $x_i - \sum_{k=0}^{D-1} a'_k x_1^k$ in the Gröbner basis of $\mathfrak{a}$ w.r.t. LEX, where $a'_0, \ldots, a'_{D-1} \in \mathbb{K}$, 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 $i = 0, \ldots, d-1$, 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 $x_i - \sum_{k=0}^{D-1} a'_k x_1^k \in \mathfrak{a}$, we have $x_i - \sum_{k=0}^{D-1} a'_k x_1^k \in \mathfrak{a} + \langle \tilde{f}_1 \rangle$, 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 $j = 0, \ldots, d-1$,

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

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 $r \leq D$ and $\deg(F_{1,i}) = d_i$. Suppose $\deg(F_1) = D$ so the ideal $\mathfrak{a}$ 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

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

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 4.1.4 we can recover efficiently the Gröbner basis of $\mathfrak{a} + \langle F_{1,i} \rangle$ 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 $\mathfrak{a} + \langle F_{1,i} \rangle$ w.r.t. LEX for $i = 1, \ldots, r$. Suppose that the Gröbner basis for i is

$$\mathcal{P}_i = [F_{1,i}, x_2 - F_{2,i}, \ldots, x_n - F_{n,i}]. \tag{4.8}$$

Then to recover the polynomials F_j in (4.1) for $j = 2, \ldots, n$, we have the following modulo equation set constructed from $\mathcal{P}_1, \ldots, \mathcal{P}_r$:

$$\begin{cases} F_j \equiv F_{j,1} \mod F_{1,1} \\ \quad \cdots \\ F_j \equiv F_{j,r} \mod F_{1,r} \end{cases}. \tag{4.9}$$

Now it is natural to give a try of the Chinese Remainder Theorem (Theorem 3.1.13).

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 $\mathfrak{a}$ is radical itself. In that case, the direct application of CRT will lead to the Gröbner basis $\mathcal{G}$ of $\mathfrak{a}$ 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 $\sqrt{\mathfrak{a}}$ 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 (4.1) 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 [91, Proposition 5(i)], and it is actually a splitting technique commonly used in the theory of triangular sets [159].

Proposition 4.1.5. *Let $\mathcal{T} \subset \mathbb{K}[\boldsymbol{x}]$ be a polynomial set in the form*

$$[R_1(x_1), x_2 - R_2(x_1), \ldots, x_n - R_n(x_1)],$$

and $R_1 = R_{1,1} \cdots R_{1,r}$. For $i = 1, \ldots, r$, define

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

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

$$\mathrm{Zero}(\mathcal{T}) = \bigcup_{i=1}^{r} \mathrm{Zero}(\mathcal{T}(i)). \tag{4.10}$$

Let $\overline{F}_1$ be the squarefree part of F_1. As each $\mathcal{P}_i$ in (4.8) satisfies the form in Proposition 4.1.5, we can compute t new polynomial sets $\overline{\mathcal{P}}_j$ 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 computes $\overline{F}_{1,j} = \gcd(F_{1,j}, P)$. As long as this polynomial is not equal to 1, a new polynomial set $\overline{\mathcal{P}}_j$ whose univariate polynomial is $\overline{F}_{1,j}$ is constructed from $\mathcal{P}_j$ by Proposition 4.1.5. Next set $P := P/\overline{F}_{1,j}$ and check whether $P = 1$. If so, we know 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 $\overline{\mathcal{P}}_1, \ldots, \overline{\mathcal{P}}_t$ to construct the modulo equation sets. Thus the Gröbner basis of $\sqrt{\mathfrak{a}}$ w.r.t. LEX can be obtained similarly (note that extracting the squarefree part of F_1 results in the radical of $\mathfrak{a}$).

The whole method based on the deterministic Wiedemann algorithm is summarized in Algorithm 11 below. The subfunction Sqrfree() returns the squarefree part of the input polynomial. The operator "cat" means concatenating two sequences.

Algorithm 11: Shape position (deterministic) $\mathcal{G}_2 := \text{ShapeDet}(\mathcal{G}_1, <_1)$

Input: $\mathcal{G}_1$, Gröbner basis of a zero-dimensional ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ w.r.t. $<_1$

Output: $\mathcal{G}_2$, Gröbner basis of $\sqrt{\mathfrak{a}}$ w.r.t. LEX if $\mathfrak{a}$ is in shape position; Fail, otherwise.

11.1 Compute the canonical basis of $\mathbb{K}[\boldsymbol{x}]/\langle\mathcal{G}_1\rangle$ and multiplication matrices $T_1, \ldots, T_n$;

11.2 $\boldsymbol{e}_1 = (1, 0, \ldots, 0)^t, \boldsymbol{e}_2 = (0, 1, 0, \ldots, 0)^t, \ldots, \boldsymbol{e}_D = (0, \ldots, 0, 1)^t \in \mathbb{K}^{(D\times 1)}$;

11.3 $k := 1$; $\mathcal{F} := [\,]$; $F; = 1$; $d := 0$; $\boldsymbol{b} = \boldsymbol{e}_1$; $S = [\,]$;

11.4 **while** $\boldsymbol{b} \neq \boldsymbol{0}$ **do**

11.5 $\quad s := [\langle \boldsymbol{e}_k, T_1^i \boldsymbol{b}\rangle : i = 0, 1, \ldots, 2(n-d)-1]$;

11.6 $\quad G := \text{BerlekampMassey}(s)$;

11.7 $\quad F := F \cdot G$; $d := \deg(F)$; $\mathcal{F} := \mathcal{F}\,\text{cat}\ [G]$; $\boldsymbol{b} := g(T_1)\boldsymbol{b}$; $S := S\,\text{cat}\ [s]$;

11.8 $\quad k := k + 1$;

11.9 **end**

11.10 (Suppose $\mathcal{F} = [F_{1,1}, \ldots, F_{1,r}]$) $F_1 := \prod_{i=1}^{r} F_{1,i}$;

11.11 **if** $\deg(F_1) \neq D$ **then**

11.12 $\quad$ **return** Fail

11.13 **else**

11.14 $\quad$ **for** $i = 1, \ldots, r$ **do**

11.15 $\quad\quad d_i := \deg(F_{1,i})$;

11.16 $\quad\quad$ **for** $j = 2, \ldots, n$ **do**

11.17 $\quad\quad\quad$ Construct the Hankel matrix H_j and the vector $\boldsymbol{b}$ from S;

11.18 $\quad\quad\quad$ Compute $\boldsymbol{c} = (c_1, \ldots, c_{d_i})^t := H_j^{-1}\boldsymbol{b}$; $f_{i,j} := \sum_{k=0}^{d_i} \boldsymbol{c}_{k+1} x_1^k$;

11.19 $\quad\quad$ **end**

11.20 $\quad$ **end**

11.21 $\quad \overline{F}_1 := \text{Sqrfree}(F_1)$;

11.22 $\quad$ **if** $\overline{F}_1 \neq F_1$ **then**

11.23 $\quad\quad$ Compute $\{[\overline{F}_{1,j}, x_2 - \overline{F}_{2,j}, \ldots, x_n - \overline{F}_{n,j}] : j = 1, \ldots, t\}$ from $\{[F_{1,i}, x_2 - F_{2,i}, \ldots, x_n - F_{n,i}] : i = 1, \ldots, r\}$ by Proposition 4.1.5 such that $\overline{F}_1 = \prod_{j=1}^{t} \overline{F}_{1,j}$ and $\overline{F}_{1,j}$ are pairwise coprime;

11.24 $\quad$ **end**

11.25 $\quad$ **for** $j = 2, \ldots, n$ **do**

11.26 $\quad\quad$ Solve the modulo equation set (4.9) to get F_j;

11.27 $\quad$ **end**

11.28 $\quad$ **return** $[\overline{F}_1, x_2 - F_2, \ldots, x_n - F_n]$

11.29 **end**

Remark 4.1.2. *If the factors $F_{1,1}, \ldots, F_{1,r}$ of F_1 returned by the deterministic Wiedemann algorithm are pairwise coprime (which needs extra computation to test), the Gröbner basis of $\mathfrak{a}$ 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 [162]. In that case, the method above with CRT *can also be used to compute the Gröbner basis of $\sqrt{\mathfrak{a}}$ w.r.t. LEX.*

Complexity

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

(1) In total the deterministic Wiedemann algorithm needs

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

operations if fast polynomial multiplications are used [162]. Here N_1 still denotes the number of nonzero entries in T_1.

(2) 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.

(3) The squarefree part $\overline{F}_1$ of F_1 can be obtained with complexity at most $O(D^2\log(D))$ for the case when $\mathbb{K}$ has characteristic 0 and $O(D^2\log(D) + D\log(q/p))$ for characteristic $p > 0$ respectively, where $|\mathbb{K}| = q$ [153, Theorem 14.20 and Exercise 14.30]. For the case when f_1 is not squarefree, suppose t new polynomial sets $\overline{\mathcal{P}_1}, \ldots, \overline{\mathcal{P}_t}$ are needed, and $\deg(\overline{F}_{1,i}) = d_i$ for $i = 1, \ldots, t$. To compute each set $\overline{\mathcal{P}}_i$ of the form (4.1), $n-1$ polynomial divisions are needed to find the remainders, with complexity $O(nd_iD)$. Hence the total complexity to obtain $\overline{\mathcal{P}}_1, \ldots, \overline{\mathcal{P}}_t$ is

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

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

(4) Solving the modulo equation set (4.9) for each $j = 2, \ldots, n$ requires $O(D^2)$ operations at most by [153, Theorem 5.7]. Thus in total $O(nD^2)$ operations are needed for the CRT application.

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

Theorem 4.1.6. *Assume that T_1 is known. If the input ideal $\mathfrak{a}$ is in shape position, then this deterministic method will return the Gröbner basis of $\sqrt{\mathfrak{a}}$ 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 $|\mathbb{K}| = q$.

Illustrative example

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

$$\mathcal{G}_1 := [x_2x_1^3 + x_1^3 + x_1 + 1, x_1^4 + x_1^3 + x_2 + 1, x_1^2 + x_2^2].$$

Its LEX Gröbner basis is

$$\mathcal{G}_2 = [F_1 := (x_1 + 1)^3(x_1^2 + x_1 + 1)^2, x_2 + x_1^4 + x_1^3 + 1],$$

from which one can see that $\mathfrak{a}$ is in shape position.

From $\mathcal{G}_1$ 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 first 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 $\mathfrak{a}$ 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 (4.4). 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 $\boldsymbol{b} = \mathbf{0}$, 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 $\mathfrak{a}$ is verified to be in shape position.

Then we construct the linear equation sets similar to (4.3) 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 $\mathfrak{a}+\langle F_{1,1}\rangle$ and $\mathfrak{a}+\langle F_{1,2}\rangle$ respectively as

$$\mathcal{P}_1 = [(x_1+1)^2(x_1^2+x_1+1), x_2+x_1],$$
$$\mathcal{P}_2 = [(x_1+1)(x_1^2+x_1+1), x_2+x_1].$$

Then the squarefree part $\overline{F}_1$ of F_1 is computed, and we find that $\mathfrak{a}$ is not radical, and thus only the Gröbner basis $\tilde{\mathcal{G}}_2$ of $\sqrt{\mathfrak{a}}$ w.r.t. LEX may be computed. From $F_{1,2} = \overline{F}_1$, we directly have $\tilde{\mathcal{G}}_2 = \mathcal{P}_2$, and the algorithm ends.

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

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

are first computed and selected according to $\overline{F}_1$ by Proposition 4.1.5. Then the modulo equation set

$$\begin{cases} F_2 \equiv x_2+1 & \mod x_1+1, \\ F_2 \equiv x_2+x_1 & \mod x_1^2+x_1+1 \end{cases}$$

as (4.9) is solved with CRT, resulting in the same $\tilde{\mathcal{G}}_2$. One can check that $\tilde{\mathcal{G}}_2$ is the Gröbner basis of $\sqrt{\mathfrak{a}}$ w.r.t. LEX with any Computer Algebra system.

4.1.3 Incremental algorithm to compute the univariate polynomial

For a zero-dimensional ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$, 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, for 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 ($O(D\log(D))$). 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 $\deg(F) = d$. 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 $O(\sum_{k=1}^{d+1} k^2) = O(d^3)$ (note that in the incremental case, the fast Berlekamp–Massey with complexity $O(k\log(k))$ is not applicable). 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, this incremental method is sensitive to the output polynomial F. When the degree d is relatively small compared with D, this method will be useful.

4.2 General ideals

In the general case when the ideal $\mathfrak{a}$ may not be in shape position, perhaps those methods described in Section 4.1 will 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.

4.2.1 Algorithm description

We first define an n-dimensional mapping $\mathrm{E} : \mathbb{Z}_{\geq 0}^n \longrightarrow \mathbb{K}$ as

$$(s_1, \ldots, s_n) \longmapsto \langle \boldsymbol{r}, T_1^{s_1} \cdots T_n^{s_n} \boldsymbol{e} \rangle, \tag{4.11}$$

where $\boldsymbol{r} \in \mathbb{K}^{(D\times 1)}$ is a random vector. One can easily 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 $\mathfrak{a}$ is of form (1.3), and the linear dependency (1.3) holds, one can verify that F satisfies (3.1) and thus is a polynomial in $\mathsf{I}(\mathrm{E})$. The BMS algorithm is precisely the one to compute the Gröbner basis of $\mathsf{I}(\mathrm{E})$ w.r.t. to a term ordering, so one may first construct the mapping E via $T_1, \ldots, T_n$, and attempts to compute the Gröbner basis of $\mathfrak{a}$ from the BMS algorithm applied to $\mathsf{I}(\mathrm{E})$.

We remark that F is in $\mathsf{I}(\mathrm{E})$ for any vector $\boldsymbol{r}$. In fact, the idea above is a multi-dimensional generalization of the Wiedemann algorithm. 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 Berlekmap–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 a 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 $\mathfrak{a}$ 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 $\mathsf{I}(\mathrm{E})$ derived from the n-dimensional mapping may lose information of $\mathfrak{a}$ because of the random vector, with $\mathfrak{a} \subset \mathsf{I}(\mathrm{E})$. Clearly, when $\mathfrak{a}$ is maximal (corresponding to the case when G in the Wiedemann algorithm is irreducible), $\mathsf{I}(\mathrm{E})$ will be equal to $\mathfrak{a}$. Furthermore, as polynomials in the Gröbner basis are characterized by the linear dependency in (1.3), we are always able to check whether the Gröbner basis of $\mathsf{I}(\mathrm{E})$ returned by the BMS algorithm is that of $\mathfrak{a}$.

Remark 4.2.1. *When the term ordering in the* BMS *algorithm is LEX, computation of the univariate polynomial in this method is exactly the same as that described in Section 4.1.1. 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* $\mathrm{E}((p_1, 0, \ldots, 0)) = \langle \boldsymbol{r}, T_1^{p_1}\boldsymbol{e} \rangle$*, and the* BMS *algorithm degenerates to the Berlekamp–Massey one.*

Another fact we would like to mention is that the BMS algorithm from Coding Theory is mainly designed for graded term orderings like DRL, for such orderings are *Archimedean* and have good properties to use in algebraic decoding [36]. But it also works for other orderings, though extra techniques not contained in the original literature have to be introduced for orderings dependent on LEX (like LEX itself or block orderings which break ties with LEX).

Take the term ordering LEX for instance, an extra polynomial reduction is performed after every BMSUpdate() step (Algorithm 9) to control the size of intermediate polynomials. This is actually not a problem for orderings like DRL, for in that case the leading term of a polynomial will give a bound on the size of terms in that polynomial. We also have to add an extra termination check for each variable x_i, otherwise

the BMS algorithm will endlessly follow a certain part of the terms. For example, all variables in the sequence $[1, x_1, x_1^2, \ldots]$ are smaller than x_2, and the original BMS does not stop handling that infinite sequence by itself.

With all the discussions, the algorithm is formulated as follows. The "Termination Criteria" here in this description mean that $\mathcal{F}$ does not change for a certain number of iterations. The subfunction Reduce($\mathcal{F}$) performs reduction on $\mathcal{F}$ so that every polynomial $F \in \mathcal{F}$ is reduced w.r.t. $\mathcal{F} \setminus \{F\}$, and IsGB($\mathcal{F}$) returns true if $\mathcal{F}$ is the Gröbner basis of $\mathfrak{a}$ w.r.t. LEX and false otherwise.

Algorithm 12: General case $\mathcal{G}_2 :=$ BMSbased$(\mathcal{G}_1, <_1)$

Input: $\mathcal{G}_1$, Gröbner basis of a zero-dimensional ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ w.r.t. $<_1$

Output: Gröbner basis of $\mathfrak{a}$ w.r.t. $<_2$; or Fail, if the BMS algorithm fails returning the correct Gröbner basis

12.1 Compute the canonical basis of $\mathbb{K}[\boldsymbol{x}]/\langle \mathcal{G}_1 \rangle$ and multiplication matrices $T_1, \ldots, T_n$;
12.2 Choose $\boldsymbol{r} \in \mathbb{K}^{(D\times 1)}$ at random;
12.3 $\boldsymbol{u} := \mathbf{0}$; $\mathcal{F} := [1]$; $\mathcal{G} := [\,]$; E := [ ];
12.4 **repeat**
12.5 $\quad e := \langle \boldsymbol{r}, T_1^{\boldsymbol{u}_1} \cdots T_n^{\boldsymbol{u}_n} \boldsymbol{e} \rangle$, E := E cat $[e]$;
12.6 $\quad \mathcal{F}, \mathcal{G} :=$ BMSUpdate$(\mathcal{F}, \mathcal{G}, \boldsymbol{u}, \mathrm{E})$;
12.7 $\quad \boldsymbol{u} :=$ Next$(\boldsymbol{u})$ w.r.t. $<_2$;
12.8 $\quad \mathcal{F} :=$ Reduce$(\mathcal{F})$;
12.9 **until** *Termination Criteria* ;
12.10 **if** IsGB$(\mathcal{F})$ **then**
12.11 $\quad$ **return** $\mathcal{F}$
12.12 **else**
12.13 $\quad$ **return** Fail
12.14 **end**

The correctness of Algorithm 12 is obvious. Next we prove its termination. Once the loop ends, the algorithm almost finishes. Hence we shall prove the termination of this loop. Clearly when the polynomial set F the BMS algorithm maintains turns to the Gröbner basis of I(E) w.r.t. $<_2$, the current termination criterion, namely $\mathcal{F}$ keeps unchanged for a certain number of passes, will be satisfied. And a sufficient condition for $\mathcal{F}$ being the Gröbner basis is given as Theorem 4.2.4 below.

4.2.2 Complexity

Part of earlier computation of values of E can be recorded to simplify the computation at Line 12.5. Suppose that the value of E at a certain term $(u_1, u_2, \ldots, u_{i-1}, u_i - 1, u_{i+1}, \ldots, u_n)$

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

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 can be achieved within $O(N)$ operations, where N is the maximal number of nonzero entries in matrices $T_1, \ldots, T_n$.

Next we focus on the case when the target term ordering is LEX. The complexities of the three steps in Algorithm 9 are analyzed below.

(1) As an extra reduction step is applied after each iteration, the numbers of terms of polynomials in $\mathcal{F}$ are bounded by $D + 1$. Denote by $\hat{N}$ the number of polynomials in $\mathcal{G}_2$. Then checking whether $\mathcal{F}$ is valid up to Next($\boldsymbol{u}$) needs $O(\hat{N}D)$ operations.

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

(3) Constructing the new polynomial set $\mathcal{F}^+$ valid up to Next($\boldsymbol{u}$) requires $O(\hat{N}D)$ operations at most. The readers may refer to [137, 36] for the way to construct new polynomials.

In step (1) above, new values of E other than e may be needed for the verification. The complexity for computing them is still $O(N)$, and this is another difference from the original BMS algorithm for graded term orderings. After the update is complete, a polynomial reduction is applied to $\mathcal{F}$ to control the size of every polynomial. This requires $O(\hat{N}\bar{N}D)$ operations, where $\bar{N}$ denotes the maximum term number of polynomials in $\mathcal{G}_2$. To summarize, the total operations needed in each pass of the main loop in Algorithm 12 is

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

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

Theorem 4.2.1. *Suppose that the input ideal* $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ *is of degree* D*. Then the number of passes of the loop in Algorithm 12 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 [17, 36] for more details.

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

Proposition 4.2.2 ([17, Corollary 9]). $\Delta(\mathrm{E}) = \{\mathrm{Span}(F) : F \notin \mathsf{I}(\mathrm{E})\}$.

Define $I(\boldsymbol{u}) := \{F \in \mathbb{K}[\boldsymbol{x}] : \mathrm{Fail}(F) > \boldsymbol{u}\}$. 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 $\boldsymbol{t} \in \mathbb{K}[\boldsymbol{x}]$.

Proposition 4.2.3 ([17, Proposition 6]). *For each* $\boldsymbol{u}$, $\Delta(\boldsymbol{u}) = \{\mathrm{Span}(F) : F \notin \mathfrak{a}(\boldsymbol{u})\}$*. Furthermore,* $\boldsymbol{v} \in \Delta(\boldsymbol{u}) \setminus \Delta(\mathrm{Pre}(\boldsymbol{u}))$ *if and only if* $\boldsymbol{v} \prec \boldsymbol{u}$ *and* $\boldsymbol{u} - \boldsymbol{v} \in \Delta(\boldsymbol{u}) \setminus \Delta(\mathrm{Pre}(\boldsymbol{u}))$*.*

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

Theorem 4.2.4 ([36, pp.529, Proposition (3.12)]). *Let* c_{max} *be the largest element of* $\Delta(\mathrm{E})$ *and* s_{max} *be the largest element of* $\{\mathrm{lt}(G) : G \in \mathcal{G}\}$*, where* $\mathcal{G}$ *is the Gröbner basis of* $\mathsf{I}(\mathrm{E})$ *w.r.t.* $<$*.*

(1) For all $\boldsymbol{u} \geq c_{max} + c_{max}$, $\Delta(\boldsymbol{u}) = \Delta(\mathrm{E})$ *holds.*

(2) For all $\boldsymbol{v} \geq c_{max} + \max\{c_{max}, s_{max}\}$*, the polynomial set* $\mathcal{F}$ *the* BMS *algorithm maintains equals* $\mathcal{G}$*.*

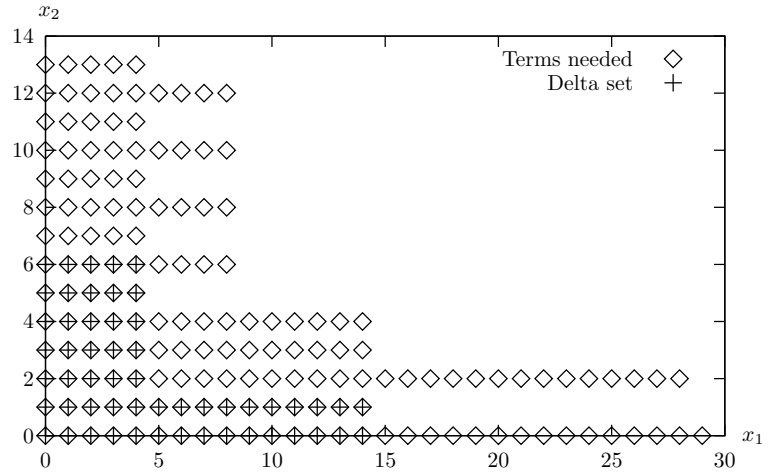

Figure 4.1: Delta set and terms needed for Cyclic5-2

As explained in Section 4.2.1, actually the term ordering LEX is not the one of interest in Coding Theory and does not possess some properties needed for a good order domain. But the results stated above are still correct. In particular, Theorem 4.2.4 indicates when the iteration in the BMS algorithm ends. For graded term orderings like DRL, once the termination term is fixed, the whole intermediate procedure in the BMS algorithm is also determined. However, for LEX it is not the case. We have to study carefully what happens between the starting term $\mathbf{1}$ and the termination term indicated by Theorem 4.2.4.

Next we first illustrate the procedure for a 2-dimensional example derived from Cyclic5. Both the delta set (marked with crosses) and the terms handled by the BMS algorithm (with diamonds) are shown in Figure 4.1.

The c_{max} and s_{max} in Theorem 4.2.4 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.

Next 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 (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.

For other rows in Figure 4.1, from Proposition 4.2.3, one knows that at a jth

diamond rows with an odd j, the delta set does not change. Thus such diamond rows are only bounded by the latest verified row in the delta set. This is because otherwise a wrong term in the delta set will be added if other diamond terms are handled. 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.

For a $2k$th diamond row, its number is related to two criteria. On one hand, again from Proposition 4.2.3, the kth cross row is determined while the $2k$th 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 $2k$th diamond row have to be handled to furnish the kth cross row. On the other hand, the number of $2k$th 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.

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 4.2.1) Suppose $\mathcal{G}$ is the Gröbner basis of $\mathsf{I}(\mathrm{E})$ 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 $\Delta_i := \Delta(\mathrm{E}) \cap \mathbb{K}[x_1, \ldots, x_i]$. From $\mathfrak{a} \subseteq \mathsf{I}(\mathrm{E})$ one knows that $\Delta(\mathrm{E})$ is a subset of the canonical basis of $\mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$, thus $|\Delta(\mathrm{E})| \leq D$. Therefore to prove the theorem, it suffices to show $2n|\Delta(\mathrm{E})|$ is an upper bound.

We induce 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 $\chi_1 \leq 2|\Delta_1|$ holds. Now suppose $\chi_k \leq 2k|\Delta_k|$ for $k(< n)$. Next we prove $\chi_{k+1} \leq 2(k+1)|\Delta_{k+1}|$.

As previously explained, in the terms to compute $\mathcal{G} \cap \mathbb{K}[x_1, \ldots, x_{k+1}]$, 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 4.2.4, we can suppose there exists some integer m, such that

$$\mathcal{T}_{k+1} = \bigcup_{l=0,\ldots,2m+1} \mathcal{T}_{k+1,l}, \quad \Delta_{k+1} = \bigcup_{j=0,\ldots,m} \Delta_{k+1,j}, \tag{4.12}$$

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 $|\mathcal{T}_{k+1,0}| = \chi_k \leq 2k|\Delta_k|$, 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 $2|\Delta_{k+1,j}|$ ($\leq 2k|\Delta_{k+1,j}|$). 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

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

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

Now we only need to show 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 $|S| \leq m$ in (4.12), 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 4.2.5. *Assume that $T_1, \ldots, T_n$ are constructed. The complexity for Algorithm 12 to complete the change of ordering is bounded by $O(nD(N + \hat{N}\bar{N}D))$, where N is the maximal number of nonzero entries in the multiplication matrices $T_1, \ldots, T_n$, and $\hat{N}$ and $\bar{N}$ are respectively the number of polynomials and the maximal term number of all polynomials in the resulting Gröbner basis.*

4.2.3 Illustrative example

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

$$\begin{aligned}\mathcal{G}_1 =\{&x_2^4 + 2\,x_1^3x_2 + 21\,x_2^3 + 11\,x_1x_2^2 + 4\,x_1^2x_2 + 22\,x_1^3 + 9\,x_2^2 + 17\,x_1x_2 + 19\,x_1^2 + \\ &2\,x_2 + 19\,x_1 + 5, x_1^2x_2^2 + 10\,x_2^3 + 12\,x_1^2x_2 + 20\,x_1^3 + 21,\ x_1^4 + 15\,x_1^2 + 19\,x_1 + 3\}.\end{aligned}$$

Here $\mathbb{F}_{65521}[x_1, x_2]/\langle \mathcal{G}_1 \rangle$ is of dimension 12. Its basis, and further the multiplication matrices T_1 and T_2, can be computed accordingly.

Now we want to compute the Gröbner basis $\mathcal{G}_2$ of $\mathfrak{a}$ w.r.t. LEX. With a vector

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

generated at random, the 2-dimensional mapping E is constructed. Then BMSUpdate() is applied term by term according to the LEX ordering, with $\Delta(\boldsymbol{u})$ and the polynomial set $\mathcal{F}$ valid up to $\boldsymbol{u}$ shown in Table 4.1. For example, at the term $(4, 0)$, the polynomial $x_1^2 + 62681\, x_1 + 41493 \in \mathcal{F}$ is not valid up to $(4, 0)$. Then the delta set is updated as $\{(0, 0), (1, 0), (2, 0)\}$, and $\mathcal{F}$ is reconstructed such that the new polynomial $x_1^3 + 62681\, x_1^2 + 35812\, x_1 + 18557$ is valid up to $(4, 0)$.

The first polynomial in $\mathcal{G}_2$:

$$G_1 = x_1^4 + 15\, x_1^2 + 19\, x_1 + 3$$

is obtained at the term $(7, 0)$. Next BMSUpdate() is executed to compute other members of $\mathsf{I}(\mathrm{E})$ 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 $\mathcal{G}_2$:

$$G_2 = x_2^3 + 7\, x_1^2x_2^2 + 15\, x_1^2x_2 + 2\, x_1^3 + 9$$

is obtained at $(3, 5)$. Now the main loop of Algorithm 12 ends. Then one can easily verify that $\{G_1, G_2\} \subset \mathcal{G}_2$ and $\dim(\mathbb{F}_{65521}[x_2, x_1]/\langle G_1, G_2 \rangle) = 12$, and thus $\mathcal{G}_2 = \{G_1, G_2\}$.

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

4.2.4 Putting all methods together: top-level algorithm

In this section, we combine the algorithms presented in the previous parts of this chapter as the following integrated top-level algorithm, which performs the change of ordering of Gröbner bases to LEX.

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
$(0,1)$	—	$g_1, x_2 + 65034\,x_1^3 + 24330\,x_1^2 + 14876\,x_1 + 52361$
$(1,1)$	—	$g_1, x_2 + 64550\,x_1^3 + 37707\,x_1^2 + 48745\,x_1 + 7628$
$(2,1)$	—	$g_1, x_2 + 38842\,x_1^3 + 5603\,x_1^2 + 45755\,x_1 + 44311$
$(3,1)$	—	$g_1, x_2 + 9449\,x_1^3 + 20826\,x_1^2 + 39078\,x_1 + 38885$
$(0,2)$	$(0,0),(1,0),(2,0),(3,0),(0,1)$	$g_1, x_2^2 + 38885\,x_2 + 65360\,x_1^3 + 1782\,x_1^2 + 36000\,x_1 + 39469$ $x_2x_1 + 20826\,x_1^3 + 28385\,x_1^2 + 55917\,x_1 + 37174$
⋮	⋮	⋮

Table 4.1: Example for the BMS-based method

Algorithm 13: Top-level algorithm $\mathcal{G}_2 := \mathrm{TopLevel}(\mathcal{G}_1, <_1)$

Input: $\mathcal{G}_1$, Gröbner basis of a zero-dimensional ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ w.r.t. $<_1$

Output: $\mathcal{G}_2$, Gröbner basis of $\mathfrak{a}$ or $\sqrt{\mathfrak{a}}$ w.r.t. LEX

```
13.1  G2 := ShapePro(G1, <1);
13.2  if G2 ≠ Fail then
13.3  |  return G2
13.4  else
13.5  |  G2 := ShapeDet(G1, <1);
13.6  |  if G2 ≠ Fail then
13.7  |  |  return G2
13.8  |  else
13.9  |  |  G2 := BMSbased(G1, <1);
13.10 |  |  if G2 ≠ Fail then
13.11 |  |  |  return G2
13.12 |  |  else
13.13 |  |  |  return FGLM(G1, <1)
13.14 |  |  end
13.15 |  end
13.16 end
```

We would like to mention that to integrate these three algorithms, one needs to skip some overlapped steps in the three algorithms, like computation of the canonical basis and the multiplication matrices, and the choice of random vectors, etc. If one

does not seek for the Gröbner basis of $\sqrt{\mathfrak{a}}$, that is to say, the multiplicities of the zeros are needed, then the deterministic invariant should be omitted.

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.

For an ideal in shape position, the probability for Algorithm 10 to compute the correct Gröbner basis is the same as that of computing the correct minimal polynomial in the Wiedemann algorithm for one choice of a random vector, which has been analyzed in [162]. When Algorithm 10 fails, the one based on the deterministic Wiedemann algorithm can tell us for sure whether the input ideal is in shape position, and return the Gröbner basis of $\sqrt{\mathfrak{a}}$. However, the probability for the BMS-based method to return the correct Gröbner basis is still unknown.

4.3 Multiplication matrices

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.

4.3.1 Construction of multiplication matrices

Given the Gröbner basis $\mathcal{G}$ of a zero-dimensional ideal $\mathfrak{a}$ w.r.t. DRL, let $B = [\epsilon_1, \ldots, \epsilon_D]$ be the canonical basis of $\mathbb{K}[\boldsymbol{x}]/\langle\mathcal{G}\rangle$, and $L := \{\mathrm{lt}(G) : G \in \mathcal{G}\}$. 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 [53].

(a) The term $\epsilon_i x_j$ is in B: the coordinate vector of $\mathrm{nform}(\epsilon_i x_j)$ 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 $G \in \mathcal{G}$ such that $\mathrm{lt}(G) = \epsilon_i x_j$;

(c) Otherwise: the normal form of $\epsilon_i x_j$ w.r.t. $\mathcal{G}$ 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 construction of the multiplication matrices with the numbers of dense columns in them.

Proposition 4.3.1. *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 [53]. □

As shown in Section 4.1, among all multiplication matrices, T_1 is the most important one, and it is also of our main interest. However, for an arbitrary ideal, now we are not able to analyze the cost to construct T_1 by isolating it from the others in Proposition 4.3.1, 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.

4.3.2 Generic sequences and Moreno-Socías conjecture

Let $\mathcal{P} = [P_1, \ldots, P_n]$ be a sequence of polynomials in $\mathbb{K}[\boldsymbol{x}]$ 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 $\mathcal{P}$ if $P_1, \ldots, P_n$

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 $\mathcal{P}$ is "a generic sequence" if its properties of our interest are generic.

For a generic sequence $[P_1, \ldots, P_n]$, its properties concerning the Gröbner basis computation, in particular the canonical basis, are the same as $[P_1^h, \ldots, P_n^h]$, where P_i^h is the homogeneous part of P_i of the highest degree. That is to say, we only need to study homogeneous generic sequences, which are also those studied in the literature. Hence in the following part of this section, a generic sequence is further assumed homogeneous.

Since we restrict to the situation where the number of polynomials is equal to that of variables, a generic sequence is a *regular* one [89]. We first recall the well-known characterization of a regular sequence via its Hilbert series.

Theorem 4.3.2. *Let $[P_1, \ldots, P_r]$ be a sequence in $\mathbb{K}[\boldsymbol{x}]$ with $\deg(P_i) = d_i$. 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 $\mathcal{P}$ be a generic sequence of degree d. Then we know its Hilbert series is

$$H(n, d) := (1 - z^d)^n/(1 - z)^n = (1 + z + z^2 + \cdots + z^{d-1})^n, \tag{4.13}$$

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

Gröbner bases of generic sequences w.r.t. DRL have been studied in [115]. A term ideal $\mathfrak{b}$ is said to be a *weakly reverse lexicographic ideal* if the following condition holds: if $\boldsymbol{t} \in \mathfrak{b}$ is a minimal generator of $\mathfrak{b}$, then $\mathfrak{b}$ contains every term of the same total degree as $\boldsymbol{t}$ which is greater than $\boldsymbol{t}$ w.r.t. some term ordering. For the term ordering DRL, we have the following conjecture due to Moreno-Socías.

Moreno-Socías conjecture ([114]) *Let $\mathbb{K}$ be an infinite field, and $\mathcal{P} = [P_1, \ldots, P_n]$*

a generic sequence in $\mathbb{K}[\boldsymbol{x}]$ *with* $\deg(P_i) = d_i$. *Then* $\mathrm{lt}(\langle\mathcal{P}\rangle)$, *the leading term ideal of* $\langle\mathcal{P}\rangle$ *w.r.t. DRL, is a weakly reverse lexicographical ideal.*

The Moreno-Socías conjecture is proven true for the codimension 2 case and for some special ideals for the codimension 3 case [1, 28]. It has been proven that this conjecture implies the Fröberg conjecture on the Hilbert series of a generic sequence, which is well-known and widely acknowledged true in the symbolic computation community [128]. Recall that for two terms $\boldsymbol{u} = (u_1, \ldots, u_n)$ and $\boldsymbol{v} = (v_1, \ldots, v_n)$, we say that $\boldsymbol{u} \prec \boldsymbol{v}$ if $u_i \leq v_i$ for $i = 1, \ldots, n$.

Proposition 4.3.3. *Use the same notations as those in the Moreno-Socías conjecture. If this conjecture holds, then for a term* $\boldsymbol{u} \in \mathrm{lt}(\langle\mathcal{P}\rangle)$, *any term* $\boldsymbol{v}$ *such that* $\deg(\boldsymbol{u}) = \deg(\boldsymbol{v})$ *and* $\boldsymbol{v} > \boldsymbol{u}$ *is also in* $\mathrm{lt}(\langle\mathcal{P}\rangle)$.

Proof. If $\boldsymbol{u}$ is a minimal generator of $\mathrm{lt}(\langle\mathcal{P}\rangle)$, then the conclusion is a direct result by the Moreno-Socías conjecture. Else there exists one minimal generator $\tilde{\boldsymbol{u}} \neq \boldsymbol{u}$ which divides $\boldsymbol{u}$. To prove the conclusion in this case, it suffices to find a term $\tilde{\boldsymbol{v}} \in \mathrm{lt}(\langle\mathcal{P}\rangle)$ which divides $\boldsymbol{v}$. Let $\tilde{\boldsymbol{v}} = \tilde{\boldsymbol{u}} - \boldsymbol{u} + \boldsymbol{v}$. If $\tilde{\boldsymbol{v}}$ is a term, by its construction we know $\deg(\tilde{\boldsymbol{v}}) = \deg(\tilde{\boldsymbol{u}})$ and $\tilde{\boldsymbol{v}} > \tilde{\boldsymbol{u}}$, and thus by the conjecture $\tilde{\boldsymbol{v}}$ is in $\mathrm{lt}(\langle\mathcal{P}\rangle)$; otherwise we may replace $\tilde{\boldsymbol{v}}$ by the biggest term $\overline{\boldsymbol{v}}$ such that $\deg(\overline{\boldsymbol{v}}) = \deg(\tilde{\boldsymbol{u}})$. Then it is easy to show that and it will work. This ends the proof. □

As Proposition 4.3.3 implies, the Moreno-Socías conjecture imposes a stronger requirement on the structure of the terms in $\mathrm{lt}(\langle\mathcal{P}\rangle)$ for a generic sequence $\mathcal{P}$. For the bivariate case, once a term $\boldsymbol{u}$ is known to be an element in $\mathrm{lt}(\langle\mathcal{P}\rangle)$, the terms in $\mathrm{lt}(\langle\mathcal{P}\rangle)$ determined by it are illustrated in Figure 4.2 (left), and furthermore, in the right figure the shape all terms in $\mathrm{lt}(\langle\mathcal{P}\rangle)$ form.

The base field in the Moreno-Socías conjecture is restricted infinite. According to our preliminary experiments on randomly generated sequences over fields of large cardinality, we find no counterexample of this conjecture. As a result, we will consider it true and use it directly. The following variant of Moreno-Socías conjecture, which is more convenient in our setting, can be derived easily from Proposition 4.3.3.

Variant of Moreno-Socías conjecture Let $\mathbb{K}$ be an infinite field, $\mathcal{P} \in \mathbb{K}[\boldsymbol{x}]$ a

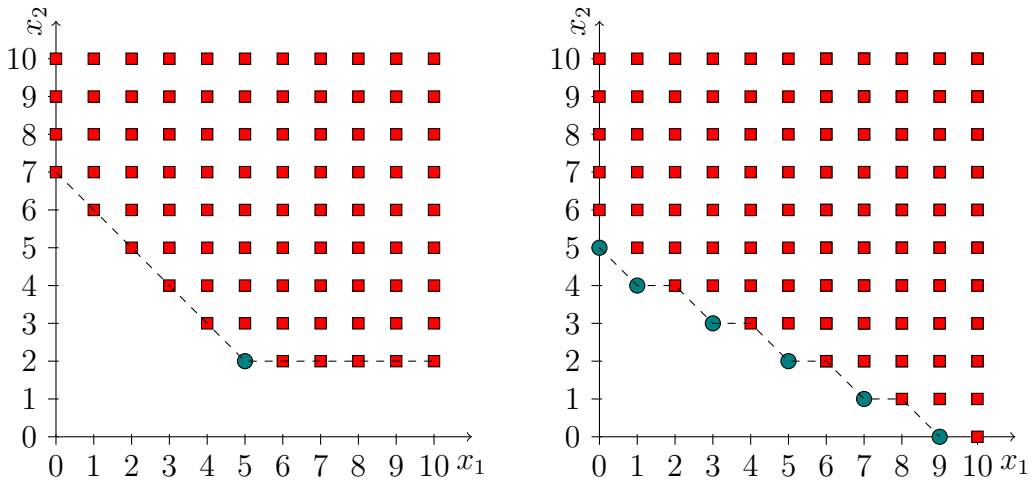

Figure 4.2: Terms and minimal generators in $\mathrm{lt}(\langle \mathcal{P} \rangle)$ for a generic sequence $\mathcal{P}$

generic sequence of degree d, and B the canonical basis of $\mathbb{K}[\boldsymbol{x}]/\langle \mathcal{P} \rangle$ 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.

4.3.3 Sparsity and construction

Let $\mathcal{P} \subset \mathbb{K}[\boldsymbol{x}]$ be a generic sequence, and $\mathcal{G}$ the Gröbner basis of $\langle \mathcal{P} \rangle$. Then polynomials in $\mathcal{G}$ 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 our variant of Moreno-Socías conjecture, we are able to give the counting of the dense columns in T_1.

Proposition 4.3.4. *Let $\mathbb{K}$, $\mathcal{P}$, B and $B(k)$ be the same as those in the Moreno-Socías conjecture variant. If the Moreno-Socías conjecture holds, 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 $\mathcal{T}^{(k)}$ be set of all terms in $\mathbb{K}[\boldsymbol{x}]$ of total degree k.

Suppose that $\boldsymbol{u}$ is the lth smallest term in $\mathcal{T}^{(k)}$. Then $x_1\boldsymbol{u}$ is still the lth smallest term in $\mathcal{T}^{(k+1)}$. Hence from the conjecture variant, if $|B(k)| \leq |B(k+1)|$, 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 4.3.3, 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 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. □

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 4.3.4, which links the canonical basis and Hilbert series.

Remark 4.3.1. *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 $d = 3$, such the greatest coefficient is called the central trinomial coefficient*.*

Corollary 4.3.5. *If the Moreno-Socías conjecture holds, then the percentage of nonzero entries in 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 4.3.4.*

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. □

Assuming the correctness of the Moreno-Socías conjecture, we can take a step forward from Proposition 4.3.4. That is, we show case (c) will not occur during the construction of T_1.

Proposition 4.3.6. *Follow the notations in the Moreno-Socías conjecture. If the conjecture holds, then for any term* $\boldsymbol{u} \notin \mathrm{lt}(\langle \mathcal{P} \rangle)$, $x_1\boldsymbol{u}$ *is either not in* $\mathrm{lt}(\langle \mathcal{P} \rangle)$ *or a minimal generator of* $\mathrm{lt}(\langle \mathcal{P} \rangle)$.

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

Without loss of generality, we can assume each $u_i \neq 0$ for $i = 1, \ldots, n$, otherwise we can reduce to the $n-1$ case by ignoring the ith component of $\boldsymbol{u}$. As $x_1\boldsymbol{u}$ is not a minimal generator, there exist a k $(1 \leq k \leq n)$ such that $\boldsymbol{u}^{(k)} := (u_1, \ldots, u_k - 1, \ldots, u_n)$ is in $\mathrm{lt}(\langle \mathcal{P} \rangle)$. The case when $k = 1$ is trivial. Otherwise, since $\deg(\boldsymbol{u}^{(k)}) = \deg(\boldsymbol{u})$ and $\boldsymbol{u}^{(k)} < \boldsymbol{u}$, by Proposition 4.3.3 we know $\boldsymbol{u} \in \mathrm{lt}(\langle \mathcal{P} \rangle)$. □

Corollary 4.3.7. *If the Moreno-Socías conjecture holds, then the number of dense columns in* T_1 *for generic sequences is equal to the cardinality of* $\{G \in \mathcal{G}_1 : x_1 | \mathrm{lt}(G)\}$, *where* $\mathcal{G}_1$ *is the Gröbner basis w.r.t. DRL.*

Remark 4.3.2. *By Corollary 4.3.7, for generic sequences, to construct* T_1 *one only needs to find the leading term of which polynomial in* $\mathcal{G}_1$ *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 4.1.2 is indeed the complete complexity for the change of ordering for generic sequences, including construction of* T_1, *the only multiplication matrix needed.*

4.3.4 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 [10, 11], where standard methods in asymptotic analysis, like the saddle-point and coalescent saddle points methods, are applied.

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 (4.13), 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 $\tilde{F}(z) := \frac{(1-z^d)^n}{(1-z)^n z^{m+1}} = e^{F(z)}$ 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 [10, Lemma 4.3.1], we have the following result.

Proposition 4.3.8. *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 [10, Section 4.3.2].

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

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

Noting that $R(1) = d$, $R'(1) = d(d-1)/2$, and $R''(1) = d(d-1)(d-2)/3$, 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 4.3.9. *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. Figure 4.3 shows the number of dense columns derived from both Proposition 4.3.4 and Corollary 4.3.9. As can be shown from this figure, the asymptotic estimation is good, even when d is small.

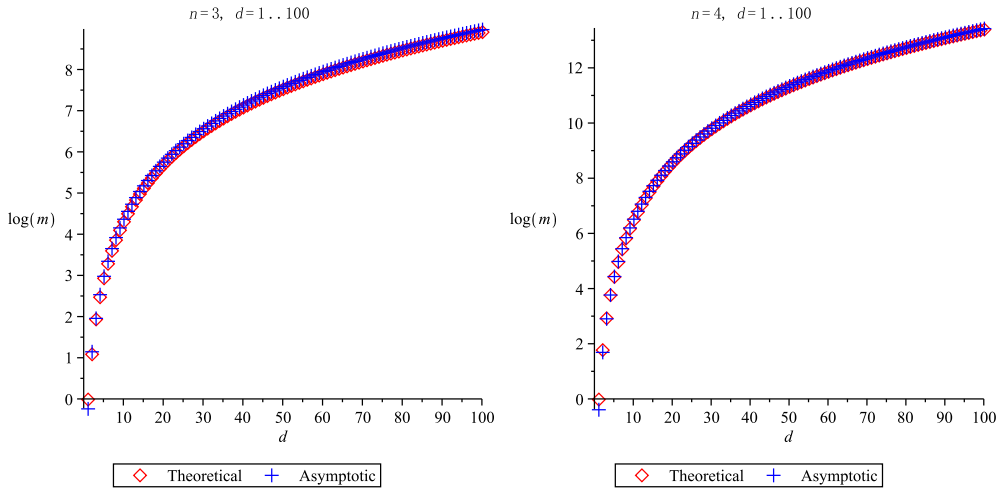

Figure 4.3: Number of dense columns in T_1 for $n = 3, 4$ and $d = 1, \ldots, 100$

Corollary 4.3.10. *Let n be fixed. As d tends to $+\infty$, if the Moreno-Socías conjecture holds, then the following statements hold:*

(a) the proportion 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 4.1.2 is $O(\sqrt{\frac{6}{n\pi}} D^{2+\frac{n-1}{n}})$.

As Corollary 4.3.10 shows, for a generic sequence, the multiplication matrix T_1 become sparser as d increases. Furthermore, the complexity of Algorithm 10 is smaller in both the exponent and constant compared with FGLM.

Remark 4.3.3. *Here we only consider the case when n is fixed and d tends to $+\infty$, while the asymptotic behaviors of the dual case when d is fixed and n tends to $+\infty$ have been studied in [11] for the special value $d = 2$.*

4.4 Implementation and experimental results

The first method for the shape position case, namely Algorithm 10, has been implemented in C over fields of characteristic 0 and finite fields. The BMS-based method for the general case has been 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, and the base field used is $\mathbb{F}_{65521}$.

Table 5.1 records the timings (in seconds) of our implementations of F_5 and Algorithm 10 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 [56] and systems coming from economic modelings (Eco n) [116, pp.148], critical points computation (Crit D, p, n) [58], and algebraic cryptanalysis of some curve-based cryptosystem (DLP EC and DLP Edwards) [52]. 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.

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 4.3.4 (column "Theoretical"), and the asymptotic one by Corollary 4.3.9 (Column "Asymp").

As shown by this table, the current implementation of Algorithm 10 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 133 and 245 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 4.3.3. 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.

		Matrix Density			FGb		Magma		Singular		Speedup	
Name	D	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%	3.3s	1.7s	18.0s	162.2s	623.9	328.6	95.4	193.3
Random 12,d=2	2^{12}	21.26%	22.56%	19.95%	19.9s	10.7s	134.9	1335.8	4867.4	2581.1	124.8	241.2
Random 13,d=2	2^{13}	19.98%	20.95%	19.16%	118.0s	80.8s	949.6	10757.4	36727.0	19820.2	133.1	245.3
Random 14,d=2	2^{14}	19.64%	20.95%	18.47%	747.2s	559.0s	7832.4	84374.6			150.9	
Random 15,d=2	2^{15}	18.52%	19.64%	17.84%	5364.6s	3894.6s						
Random 4,d=8	4096	7.54%	8.40%	8.64%	2.5s	4.2s	3.5s	556.7s			132.5	
Random 4,d=9	6561	6.66%	7.45%	7.68%	6.6s	13.5s	9.3s	1800.9s			133.4	
Random 4,d=10	10000	5.97%	6.70%	6.91%	15.5s	45.7s						
Random 4,d=11	14641	5.40%	6.09%	6.28%	36.8s	127.0s						
Random 4,d=12	20736	4.94%	5.57%	5.76%	81.3s	363.6s						
Random 4,d=13	28561	4.55%	5.14%	5.32%	182.0s	713.9s						
Random 3,d=15	3375	4.46%	5.01%	5.32%	0.7s	1.8s	0.95s	262.9s			146.1	
Random 3,d=16	4096	4.17%	4.69%	4.99%	1.1s	2.9s	1.5s	333.8s			115.1	
Random 3,d=17	4913	3.92%	4.42%	4.69%	1.5s	4.4s	1.95s	585.3s			133.0	
Random 3,d=18	5832	3.70%	4.17%	4.43%	2.3s	6.5s	2.9s	1142.6s			175.8	
Random 3,d=20	8000	3.32%	3.75%	3.99%	4.3s	13.9s						
Random 3,d=22	10648	3.01%	3.41%	3.63%	8.1s	27.9s						
Random 3,d=24	13824	2.76%	3.13%	3.32%	14.1s	57.0s						
Random 3,d=30	27000	2.20%	2.50%	2.66%	65.9s	325.7s						
Random 3,d=40	64000	1.64%	1.88%	1.99%	470.1s	3491.5s						
Katsura11	2^{11}	21.53%	22.56%	20.83%	4.7s	1.7s	18.3s	178.6s	632.0s	328.4s	105.1	193.2
Katsura12	2^{12}	21.26%	22.56%	19.95%	29.6s	10.6s	147.9s	1408.1s	5061.8s	2623.5s	132.8	247.5
Katsura13	2^{13}	19.86%	20.95%	19.16%	177.2s	80.9s	1037.2s	10895.4s			134.7	
Katsura14	2^{14}	19.64%	20.95%	18.47%	1285.1s	553.4 s	9599.0s	87131.9s			157.4	
Katsura15	2^{15}	18.52%	19.64%	17.84%	8487.0s	3874.6s						
Eco 12 ‡	1024	29.69%	NA	NA	2.8s	0.35s	29.6s	456.5s			1304	
Eco 13 ‡	2048	27.52%	NA	NA	15.3s	2.0s	262.7s	5692.7s			2846	
Eco 14 ‡	4096	26.4%	NA	NA	85.2s	13.0s						
Eco 15 ‡	8192	24.9%	NA	NA	587.0s	96.3s						
Crit D=2,p=4,n=9	896	30.17%	NA	NA	0.5s	0.24s	3.0s	17.4s			72.5	
Crit D=2,p=4,n=10	1344	31.13%	NA	NA	1.6s	0.68s	11.8s	62.1s			91.3	
Crit D=2,p=4,n=11	1920	31.86%	NA	NA	4.3s	1.77s	40.7s	192.5s			108.8	
Crit D=3,p=3,n=6	2160	17.52%	NA	NA	1.3s	1.4s	4.3s	134.2s			95.9	
Crit D=3,p=3,n=7	6480	17.39%	NA	NA	26.2s	34.2s	122.8s	4139.4s			121	
Crit D=3,p=3,n=8	18144	17.63%	NA	NA	520.1s	762.6s						
Crit D=4,p=2,n=5	1728	14.46%	NA	NA	0.48s	0.70s	1.4s	57.3s			81.9	
Crit D=4,p=2,n=6	6480	14.11%	NA	NA	16.3s	27.3s	63.2s	3196.7s			117.1	
Crit D=5,p=2,n=5	6400	11.00%	NA	NA	10.2s	19.9s	27.9s	2335.1s			117.3	
Crit D=6,p=2,n=5	18000	8.80%	NA	NA	123.4s	346.7s						
MinR(9,6,3)	980	26.82%	NA	NA	0.7s	0.3s	6.3s	22.7s	137.5s	38.1s	75.7	127.0
MinR(9,7,4)	4116	22.95%	NA	NA	17.5s	10.8s	208.1s	1360.4s	4985.8s	2490.3s	126.0	230.6
MinR(9,8,5)	14112	19.04%	NA	NA	297.4s	337.7s						
MinR(9,9,6)	41580	16.91%	NA	NA	5010.7s	7086.6s						
DLP EC n=4	4096	7.54%	NA	NA	2.4s	4.25s	7.4s	475.8s	72.4s	1823.6s	112.0	429.1
DLP Edwards n=4	512	7.68%	NA	NA	0.03s	0.02s	0.16s	21.4s			1066	
DLP Edwards n=5	2^{16}	3.30%	NA	NA	1802.0s	5742.7s						
Cyclic 10 †	34940	1.00%	NA	NA		1107.8s	>16 hrs and >16 Gig					

Table 4.2: Timings for Algorithm 10 for the shape position case from DRL to LEX

Table 4.3 illustrates the performances of Algorithm 12 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. Besides n and D denoting the number of variables and degree of the input ideal, the columns "Mat Density" and "Poly Density" denote the maximal percentage of nonzero entries in the matrices $T_1, \ldots, T_n$ and the density of resulting Gröbner bases respectively. The following 4 columns record the numbers of passes in the main loop of Algorithm 12, matrix multiplications, reductions and field multiplications.

As one can see from this table, the numbers of passes accord with the bound derived in theorem 4.2.1, 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 13, it is not common to handle such ideals in shape position with this method.

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	$nD^{2.968}$

Table 4.3: Performances of Algorithm 12 for the general case from DRL to LEX

Chapter 5

Simple decomposition over finite fields

As stated in Section 2.3.2, simple sets are a special kind of regular sets in which every polynomial is conditionally *squarefree* with respect to its leading variable. One of its remarkable properties is that the number of its zeros (all of multiplicity 1) in the algebraic closure of the ground field $\mathbb{K}$ can be easily counted by looking at the leading degrees of its polynomials. Compared with an irreducible triangular set which defines a finite extension of $\mathbb{K}$, a simple set may represent a product of such field extensions. The representation of field extensions is clearly made more compact by using one simple set than using several irreducible triangular sets. Compactness is desirable when produced triangular sets need be manipulated in further computations, for otherwise the size of intermediate results may grow out of control.

Algorithms for decomposing polynomial sets into simple sets have been proposed by Wang [157] and by Bächler and others [8] for the case in which $\mathbb{K}$ is of characteristic 0. However, To our knowledge, there is no algorithm available for simple decomposition over finite fields, for squarefree decomposition of polynomials become more complicated in this case. The purpose of this chapter is to propose original algorithms for simple decomposition over finite fields.

The results in this chapter are based on the joint work with Dongming Wang and Xiaoliang Li and are published as [98, 118].

5.1 Simple sets revisited

In Section 2.3.2, we have defined simple sets in the geometric way (by regular zeros). In this section we revisit simple sets and their properties in an algebraic way (mainly by ideals).

Recall that for any triangular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$, $\boldsymbol{u} = (u_1, \ldots, u_{n-r})$ and $\boldsymbol{y} = (y_1, \ldots, y_r)$ are its parameters and dependents respectively, and a variable ordering $u_1 < \cdots < u_{n-1} < y_1 < \cdots < y_r$ is assumed. We will work extensively on the transcendental extension $\tilde{\mathbb{K}} := \mathbb{K}(\boldsymbol{u})$ of $\mathbb{K}$. For the clarity, for an ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ we write $\mathfrak{a}_{\tilde{\mathbb{K}}}$ to indicate the extension of $\mathfrak{a}$ in $\tilde{\mathbb{K}}[\boldsymbol{y}]$.

Let $\mathcal{T} = [T_1, \ldots, T_r]$ be a regular set and F a polynomial in $\mathbb{K}[\boldsymbol{u}][\boldsymbol{y}_i]$. Then for any prime ideal $\mathfrak{p} \subset \tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]$, the image of F in $(\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\mathfrak{p})[y_i]$ under the natural homomorphism is denoted by $\overline{F}^{\mathfrak{p}}$. Similarly for any polynomial set $\mathcal{P} \subset \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}_i]$, $\overline{\mathcal{P}}^{\mathfrak{p}} := \{\overline{P}^{\mathfrak{p}} : P \in \mathcal{P}\}$. We write $\overline{F}$ for the image of F in $(\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\sqrt{\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}})[y_i]$, and $\overline{\mathcal{P}}$ in a similar way. In the case $i = 1$, $\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]$ and $\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$ are understood as $\tilde{\mathbb{K}}$ and $\{0\}$ respectively. The notations $\overline{F}$ and $\overline{\mathcal{P}}$ rarely incur ambiguity, for we usually work on only one regular set $\mathcal{T}$.

Suppose that $\mathfrak{p}_1, \ldots, \mathfrak{p}_s$ are all the associate primes of $\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$. From Proposition 2.3.2((b)), one knows that for all $j \neq k$, $\mathfrak{p}_j \not\subset \mathfrak{p}_k$, and thus $\mathfrak{p}_j + \mathfrak{p}_k = \langle 1 \rangle$. Hence by the Chinese Remainder Theorem (Theorem 3.1.13)

$$\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\sqrt{\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}} \cong \prod_{j=1}^{s} \tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\mathfrak{p}_j. \tag{5.1}$$

Furthermore, the ideal $\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$ is zero-dimensional in $\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]$, and thus each $\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\mathfrak{p}_j$ is a field. In other words, $\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\sqrt{\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}}$ is isomorphic to a product of fields. In particular,

$$(\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\sqrt{\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}})[y_i] \cong \prod_{j=1}^{s} (\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\mathfrak{p}_j)[y_i].$$

In what follows, we do not distinguish between the corresponding elements in the two rings on the two sides of the isomorphism above. With the notations introduced previously, we have $\overline{F} = (\overline{F}^{\mathfrak{p}_1}, \ldots, \overline{F}^{\mathfrak{p}_s})$ for any $F \in \tilde{\mathbb{K}}[\boldsymbol{y}_i]$ and define the natural projection $\pi_j : (\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\sqrt{\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}})[y_i] \to (\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\mathfrak{p}_j)[y_i]$ as $\pi_j(\overline{F}) := \overline{F}^{\mathfrak{p}_j}$ for $j = 1, \ldots, s$.

Definition 5.1.1. A regular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$ with $\mathrm{lv}(T_i) = y_i$ $(i = 1, \ldots, r)$ is called a *simple set* if for any i and associated prime $\mathfrak{p}$ of $\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$, the polynomial $\overline{T}_i^{\mathfrak{p}}$ is squarefree in $(\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\mathfrak{p})[y_i]$.

For example, in $\mathbb{F}_5[u, x, y]$ with $u < x < y$, $[(x^2 - u)(x^3 - u),\ y^2 + 2\,xy + u]$ is not a simple set (because the image of $y^2 + 2\,xy + u$ in $(\tilde{\mathbb{F}}_5[x]/\langle x^2 - u\rangle)[y]$ is $(y + x)^2$, which is not squarefree), but $[(x^2 - u)(x^3 - u),\ y^2 + u]$ is. One can see that as representations of the same field extensions the above simple set is more compact than the two irreducible triangular sets $[x^2 - u,\ y^2 + u]$ and $[x^3 - u,\ y^2 + u]$.

The concept of simple sets was derived by Wang [157] from Thomas' classical concept of simple systems [149] and in the tutorial article [74] by Hubert simple sets are called *squarefree regular chains*.

One can show that the definition above of simple sets via associate primes is equivalent to the one via regular zeros. Roughly speaking, this is because in some sense a regular zero of $\mathcal{T}_{\leq i-1}$ is the zero of $\mathcal{T}_{\leq i-1} = 0$ over $\mathbb{K}(\boldsymbol{u})$, where $\boldsymbol{u}$ are all the transcendental elements (See Definition 2.3.2). Hence, substituting a regular zero in T_i is equivalent to projecting T_i to $\overline{T}_i^{\mathfrak{p}}$ for the corresponding associated prime $\mathfrak{p}$ of $\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$ for that regular zero. The most remarkable property of simple sets is as stated in Theorem 5.1.3 below.

Lemma 5.1.1. *Let* $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$ *be a regular set with* $\mathrm{lv}(T_i) = y_i$ $(1 \leq i \leq r)$. *Then* $\overline{\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}} = \langle \overline{T}_i \rangle$.

Proof. By Proposition 2.3.3, $\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}} = \langle \mathcal{T}_{\leq i} \rangle_{\tilde{\mathbb{K}}}$. For any $T \in \mathcal{T}_{\leq i-1}$, we have $T \in \mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}} \subseteq \sqrt{\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}}$ and thus $\overline{T} = 0$. The conclusion follows. □

Lemma 5.1.2. *Let* $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$ *be a regular set and* $F \in \mathbb{K}[\boldsymbol{x}]$ *a polynomial with* $\mathrm{lv}(F) = y_i$ $(1 \leq i \leq r)$. *If* $\overline{F} \in \langle \overline{T}_i \rangle$ *and* $\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$ *is radical, then* $F \in \mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$.

Proof. From the pseudo-division of F w.r.t. $\mathcal{T}_{\leq i}$, we get $MF = \sum_{j=1}^{i-1} Q_j T_j + Q_i T_i + R$, where $R = \mathrm{prem}(F, \mathcal{T}_{\leq i})$, $M = \prod_{j=1}^{i} \mathrm{ini}(T_j)^{s_j}$, and s_j is a nonnegative integer. Hence $\overline{M}\,\overline{F} = \overline{Q}_i\,\overline{T}_i + \overline{R}$. If $\overline{F} \in \langle \overline{T}_i \rangle$, then $\overline{R} \in \langle \overline{T}_i \rangle$. Noting that $\deg(\overline{R}, y_i) < \deg(\overline{T}_i, y_i)$, we have $\overline{R} = 0$.

Write $R = \sum_{l=0}^{d} A_l y_i^l$, where $A_l \in \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}_{i-1}]$. Then $\overline{R} = \sum_{l=0}^{d} \overline{A}_l y_i^l = 0$. It follows that $\overline{A}_l = 0$ for all $l = 0, \ldots, d$. Therefore, $A_l \in \sqrt{\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}} = \mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$, and thus $A_l \in \mathrm{sat}_{i-1}(\mathcal{T})$. By Proposition 2.3.1, we have $A_l = \mathrm{prem}(A_l, \mathcal{T}_{<i}) = 0$. Hence $R = 0$ and $F \in \mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$. □

Theorem 5.1.3. *For any regular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$, the following statements are equivalent:*

(a) $\mathcal{T}$ is simple;

(b) $\mathrm{sat}(\mathcal{T})$ is radical;

(c) $\mathrm{sat}(\mathcal{T})_{\tilde{\mathbb{K}}}$ is radical;

(d) $\mathrm{sat}_i(\mathcal{T})$ is radical for all $i = 1, \ldots, r$;

(e) $\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$ is radical for all $i = 1, \ldots, r$.

Proof. (a) $\Longleftrightarrow$ (e). Obviously, (a) implies (e) for $i = 1$. In order to apply induction to i, supposing that $\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$ is radical, we need to prove that $\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$ is also radical. As $\mathcal{T}$ is simple, for any associated prime $\mathfrak{p}$ of $\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$, $\overline{T}_i^{\mathfrak{p}}$ is a squarefree polynomial and thus $\langle \overline{T}_i^{\mathfrak{p}} \rangle$ is radical in $(\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\mathfrak{p})[y_i]$. Hence $\langle \overline{T}_i \rangle$ is radical in $(\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}})[y_i]$. If $\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$ is not radical, then there exist a polynomial $G \in \tilde{\mathbb{K}}[\boldsymbol{y}_i]$ and some integer k such that $G^k \in \mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$, but $G \notin \mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$. Therefore, we have $(\overline{G})^k \in \langle \overline{T}_i \rangle$ and $\overline{G} \notin \langle \overline{T}_i \rangle$ by Lemmas 5.1.1 and 5.3.5. This is a contradiction.

If $\mathcal{T}$ is not simple, then let i be the smallest integer such that the condition in the definition of simple set does not hold, i.e. there exists an associated prime $\mathfrak{p}$ of $\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$ (radical) such that $\overline{T}_i^{\mathfrak{p}}$ is not squarefree. In this case, $\overline{\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}}^{\mathfrak{p}} = \langle \overline{T}_i^{\mathfrak{p}} \rangle$ is not radical. Hence $\langle \overline{T}_i \rangle$ is not radical, and neither is $\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$.

(e) $\Longleftrightarrow$ (d). If $\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$ is radical, then obviously $\mathrm{sat}_i(\mathcal{T})$ is also radical as $\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}} \cap \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}_i] = \mathrm{sat}_i(\mathcal{T})$.

Suppose that $\mathrm{sat}_i(\mathcal{T})$ is radical and recall that $\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}} = (\mathbb{K}[\boldsymbol{u}] \setminus \{0\})^{-1} \mathrm{sat}_i(\mathcal{T})$. For any $(F/G)^s = F^s/G^s \in \mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$, we have $F^s \in \mathrm{sat}_i(\mathcal{T})$ and $G^s \in \mathbb{K}[\boldsymbol{u}]$; hence $F \in \mathrm{sat}_i(\mathcal{T})$ and $G \in \mathbb{K}[\boldsymbol{u}]$. Consequently, $F/S \in \mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$.

The proof of (b) $\Longleftrightarrow$ (d) and (c) $\Longleftrightarrow$ (e) is trivial, as $\mathrm{sat}_i(\mathcal{T}) = \mathrm{sat}(\mathcal{T}) \cap \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}_i]$ and $\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}} = \mathrm{sat}(\mathcal{T})_{\tilde{\mathbb{K}}} \cap \tilde{\mathbb{K}}[\boldsymbol{y}_i]$. □

The above property that the saturated ideal of a simple set is radical is consistent to the definition that each polynomial in a simple set is squarefree modulo the proceeding saturated ideal. By the isomorphism (5.1), we know that for a simple set $\mathcal{S}$, $\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\operatorname{sat}_{i-1}(\mathcal{S})_{\tilde{\mathbb{K}}}$ is indeed a product of field extensions of $\tilde{\mathbb{K}}$, and therefore of $\mathbb{K}$. This structure should be kept in mind for the readers to better understand our study on simple sets. We can see that simple sets are good representations of products of field extensions, which is a key idea we will further exploit in Chapter 6.

The main purpose of this chapter is to present algorithms for decomposing any polynomial set $\mathcal{F} \subset \mathbb{F}_q[\boldsymbol{x}]$ for a finite field $\mathbb{F}_q$ into finitely many simple sets $\mathcal{S}_1, \ldots, \mathcal{S}_k$ such that $\sqrt{\langle \mathcal{F} \rangle} = \bigcap_{i=1}^{k} \operatorname{sat}(\mathcal{S}_i)$ holds. Such a decomposing process is called *simple decomposition*. With the properties of the decomposed simple sets, we are able to study $\sqrt{\mathcal{F}}$ (or $\mathsf{V}(\mathcal{F})$ with the ideal-variety correspondence) by simple sets.

Since algorithms exist for decomposing $\mathcal{F}$ into regular sets $\mathcal{T}_1, \ldots, \mathcal{T}_s$ such that $\sqrt{\mathcal{F}} = \bigcap_{i=1}^{s} \sqrt{\operatorname{sat}(\mathcal{T}_i)}$, which are effective over fields of both characteristics 0 and p (name such an algorithm as $\{\mathcal{T}_1, \ldots, \mathcal{T}_s\} = \mathsf{RegDec}(\mathcal{F})$ for later use) [77, 74], to design algorithms for simple decomposition over finite fields we only need to study how to turn a regular set into one or several simple sets. This turning problem can be further reduced by induction to the following relatively easier problem.

Problem. Given a regular set $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{F}_q[\boldsymbol{x}]$, with $[T_1, \ldots, T_{r-1}]$ being a simple set, compute finitely many simple sets $\mathcal{S}_1, \ldots, \mathcal{S}_k \subset \mathbb{F}_q[\boldsymbol{x}]$ such that

$$\sqrt{\operatorname{sat}(\mathcal{T})} = \bigcap_{i=1}^{k} \operatorname{sat}(\mathcal{S}_i).$$

One natural way to solve this problem is to decompose the radical ideal $\sqrt{\operatorname{sat}(\mathcal{T})}$ into prime components, each component corresponding to an irreducible triangular set (which is a special simple set $\mathcal{S}_i$). Efficient algorithms for prime decomposition over finite fields have been well studied and implemented [124, 144] (for example, prime decompositions for most of the examples in Table 5.5 can be computed in the matter of seconds by using primedec_mod in the Risa/Asir[1] system [122]). However, prime decomposition splits products of field extensions (determined by simple sets) completely and needs irreducible factorization of polynomials (which is an expensive

[1] http://www.math.kobe-u.ac.jp/Asir/

process). We are interested in simple decomposition because it keeps products of field extensions with as little splitting as possible, thus having its representation usually more compact than the representation of prime decomposition, and it only needs generalized gcd computation rather than irreducible factorization.

In the following two sections, simple decomposition for the two cases of zero-dimensional and positive-dimensional polynomial sets is discussed respectively.

5.2 Zero-dimensional polynomial sets

There exist several effective algorithms for simple decomposition over fields of characteristic 0 like $\mathbb{Q}$ [77, 74]. These algorithms all make use of the property that for a univariate polynomial F over a field of characteristic 0, $F/\gcd(F, F')$ is its squarefree part. However, as already explained in Section 3.2.2, this property does not hold any longer for finite fields, and the use of $F/\gcd(F, F')$ will lose some factor in the squarefree part of F. For example, for the polynomial $F = x^3(x-1)^2 \subset \mathbb{F}_3[x]$, we have $F' = 2x^3(x-1)$, and thus $\gcd(F, F') = x^3(x-1)$. Then $F/\gcd(F, F') = x - 1$, causing the squarefree factor x of F lost.

One may notice that the factor x^3 of F is where the problem comes, for $(x^3)' = 0$. In fact, for a field of characteristic p, any pth power will behave in this way. Therefore to compute the squarefree part of a polynomial, it is necessary to isolate its factors as pth powers and extract their pth roots. We fist study this problem for the relatively easier zero-dimensional case.

The major difference of the case of zero-dimensional polynomial sets from positive ones is that the field we are working on is indeed an algebraic extension of a finite field, and thus a perfect field. We start our algorithm for simple decomposition of zero-dimensional polynomial sets over finite fields with recalling perfect fields and their properties.

5.2.1 Perfect field

A *perfect field* is a field for which any algebraic extension is separable. As a simple criterion, a field $\mathbb{K}$ is perfect if and only if

(a) $\mathbb{K}$ is of characteristic 0, or

(b) $\mathbb{K}$ is of characteristic $p > 0$, and every element of $\mathbb{K}$ has a pth root in $\mathbb{K}$.

In particular, every finite field is perfect, and any algebraic extension of a perfect field keeps to be perfect. As shown by (b), a field of characteristic $p > 0$ is perfect if and only if the pth roots are extractable for all its elements. In fact, pth root extraction is one important problem of our later interest.

For a perfect field, we may derive the following properties useful for checking squarefreeness.

Lemma 5.2.1. *Let $\mathbb{K}$ be a perfect field and $P \in \mathbb{K}[x]$. Then P is squarefree if and only if P' is regular modulo $\langle P \rangle$.*

Proposition 5.2.2. *Let $\mathcal{T} = [T_1, \ldots, T_r]$ be a regular set in $\mathbb{K}[\boldsymbol{x}]$. If $\tilde{\mathbb{K}}$ is a perfect field, then $\mathcal{T}$ is simple if and only if for all $S \in \operatorname{sep}(\mathcal{T})$, S is regular modulo $\operatorname{sat}(\mathcal{T})_{\tilde{\mathbb{K}}}$.*

Proof. ($\Rightarrow$) Suppose that $\mathcal{T}$ is a simple set and $S = \operatorname{sep}(T_i)$ is not regular modulo $\operatorname{sat}(\mathcal{T})_{\tilde{\mathbb{K}}}$ for some $i \in \{1, \ldots, r\}$. Then S is not regular modulo $\operatorname{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$. Hence there exists an $F \in \tilde{\mathbb{K}}[\boldsymbol{y}_i]$ such that $SF \in \operatorname{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$, but $F \notin \operatorname{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$. Thus by Lemma 5.1.2 there exists some associated prime $\mathfrak{p}$ of $\operatorname{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$ such that $\overline{F}^{\mathfrak{p}} \notin \overline{\operatorname{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}}^{\mathfrak{p}}$, and by Lemma 5.1.1 we have $\overline{S}^{\mathfrak{p}}\overline{F}^{\mathfrak{p}} \in \overline{\operatorname{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}}^{\mathfrak{p}} = \langle \overline{T}_i^{\mathfrak{p}} \rangle$. Therefore, $\overline{S}^{\mathfrak{p}}$ is not regular modulo $\langle \overline{T}_i^{\mathfrak{p}} \rangle$, and by Lemma 5.2.1, $\overline{T}_i^{\mathfrak{p}}$ is not squarefree. This leads to a contradiction, so every $S \in \operatorname{sep}(\mathcal{T})$ must be regular modulo $\operatorname{sat}(\mathcal{T})_{\tilde{\mathbb{K}}}$.

($\Leftarrow$) For any $T_i \in \mathcal{T}$, suppose that $S_i = \operatorname{sep}(T_i)$ is regular modulo $\operatorname{sat}(\mathcal{T})_{\tilde{\mathbb{K}}}$ and thus regular modulo $\operatorname{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$. Then for any associated prime $\mathfrak{p}$ of $\operatorname{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$, $\overline{S}_i^{\mathfrak{p}}$ is regular modulo $\langle \overline{T}_i^{\mathfrak{p}} \rangle$. By Lemma 5.2.1, $\overline{T}_i^{\mathfrak{p}}$ is squarefree. Consequently, $\mathcal{T}$ is simple.□

Corollary 5.2.3. *Let $\mathcal{T}$ be a regular set in $\mathbb{K}[\boldsymbol{x}]$. If $\tilde{\mathbb{K}}$ is a perfect field, then $\mathcal{T}$ is simple if and only if S is regular modulo $\operatorname{sat}(\mathcal{T})$ for all $S \in \operatorname{sep}(\mathcal{T})$.*

If the field $\tilde{\mathbb{K}}$ is not perfect, then the conclusions of Proposition 5.2.2 and Corollary 5.2.3 do not necessarily hold. For example, $[y^3 - u] \subseteq \mathbb{F}_3[u, y]$ is a simple set, but $\operatorname{sep}(y^3 - u) = 0$ (under $u < y$). As fields of characteristic 0 are perfect, the equivalent condition in Corollary 5.2.3 may be used to define simple sets over the field of rational numbers (see, e.g., [74]).

5.2.2 Generalized squarefree decomposition

In this part we generalize the algorithm for squarefree decomposition of a univariate polynomial to the case modulo a zero-dimensional simple set.

Definition 5.2.1. For any zero-dimensional simple set $\mathcal{S} \subseteq \mathbb{K}[\boldsymbol{x}]$ and polynomial $F \in \mathbb{K}[\boldsymbol{x}][z] \setminus \mathbb{K}[\boldsymbol{x}]$, the *generalized squarefree decomposition* of F w.r.t. $\mathcal{S}$ is a set $\{(\{[F_{i1}, a_{i1}], \ldots, [F_{ik_i}, a_{ik_i}]\}, \mathcal{S}_i) : i = 1, \ldots, s\}$ such that

(a) Each $\mathcal{S}_i$ is a simple set in $\mathbb{K}[\boldsymbol{x}]$ for $i = 1, \ldots, s$, and $\operatorname{sat}(\mathcal{S}) = \operatorname{sat}(\mathcal{S}_1) \cap \cdots \cap \operatorname{sat}(\mathcal{S}_s)$ is an irredundant decomposition;

(b) each $\{[\overline{F}^{\mathfrak{p}}_{i1}, a_{i1}], \ldots, [\overline{F}^{\mathfrak{p}}_{ik_i}, a_{ik_i}]\}$ is the squarefree decomposition of $\overline{F}^{\mathfrak{p}}$ for any associated prime $\mathfrak{p}$ of $\operatorname{sat}(\mathcal{S}_i)$.

Though $\mathbb{K}[\boldsymbol{x}]/\operatorname{sat}(\mathcal{S})$ is not a UFD, we reduce the definition of generalized squarefree decomposition to each field $\mathbb{K}[\boldsymbol{x}]/\mathfrak{p}$. The critical idea behind this concept is that we combine ordinary squarefree decompositions of $\overline{F}^{\mathfrak{p}}$ for different $\mathfrak{p}$s if the decompositions are of the same form. This idea is further studied in Chapter 6.

In the following extended algorithm, the extraction of pth root of an element modulo a simple set is an important ingredient and we will discuss it later. The operation $\mathsf{pop}(\mathbb{D})$ in the algorithm below means to take one element randomly and then delete it from $\mathbb{D}$.

Algorithm 14: Generalized squarefree decomposition $\mathbb{S} := \mathsf{sqfZero}(F, \mathcal{S})$

Input: F — a polynomial in $\mathbb{F}_q[\boldsymbol{x}][z] \setminus \mathbb{F}_q[\boldsymbol{x}]$;
$\mathcal{S}$ — a zero-dimensional simple set in $\mathbb{F}_q[\boldsymbol{x}]$.

Output: $\mathbb{S}$ — the generalized squarefree decomposition of F w.r.t. $\mathcal{S}$.

```
14.1  𝕊 := ∅; 𝔻 := ∅;
14.2  for (C_1, 𝒞) ∈ pgcd({F, sep(F)}, 𝒮) do
14.3      B_1 := F/C_1  mod 𝒞;
14.4      𝔻 := 𝔻 ∪ {[B_1, C_1, 𝒞, ∅, 1]};
14.5  end
14.6  while 𝔻 ≠ ∅ do
14.7      [B_1, C_1, 𝒞, ℙ, d] := pop(𝔻);
14.8      if deg(B_1, z) > 0 then
14.9          for (B_2, 𝒜) ∈ pgcd({B_1, C_1}, 𝒞) do
14.10             C_2 := C_1/B_2  mod 𝒜;
14.11             P := B_1/B_2  mod 𝒜;
14.12             if deg(P, z) > 0 then ℙ := ℙ ∪ {[P, d]};
14.13             𝔻 := 𝔻 ∪ {[B_2, C_2, 𝒜, ℙ, d + 1]};
14.14         end
14.15     else
14.16         if deg(C_1, z) > 0 then
14.17             C_3 := the pth root of C_1 in (𝔽_q[x]/ sat(𝒞))[z];
14.18             for ({[F_1, a_1], ..., [F_s, a_s]}, ℬ) ∈ sqfZero(C_3, 𝒞) do
14.19                 𝕊 := 𝕊 ∪ {(merge({[F_1, a_1 · p], ..., [F_s, a_s · p]}, ℙ), ℬ)};
14.20             end
14.21         else
14.22             𝕊 := 𝕊 ∪ {(ℙ, 𝒞)};
14.23         end
14.24     end
14.25 end
```

In this algorithm, $\mathbb{D}$ stores what to be processed. For each element $[B, C, \mathcal{C}, \mathbb{P}, d] \in \mathbb{D}$, $\mathcal{C}$ is a simple set over which later computation is to be performed; $\mathbb{P}$ stores the squarefree components already obtained, which are of power smaller than d.

Proof. (Termination) Suppose that the call $\mathsf{sqfZero}(C, \mathcal{S})$ terminates for any polynomial C whose degree in z is smaller than $\deg(F, z)$ by induction. As the **while** loop is essentially a splitting procedure, we can regard it as building trees with elements in $\mathbb{D}$ as their nodes. The roots of these trees are built in Lines 14.2–14.5. For each node $[B_1, C_1, \mathcal{C}, \mathbb{P}, d]$, its child $[B_2, C_2, \mathcal{A}, \mathbb{P}, d+1]$ is constructed in Line 14.13 with $\deg(B_2, z) < \deg(B_1, z)$ and the parameter d indicates its depth in the trees. Hence each path of the trees must be finite. The termination of Line 14.18, where sqfZero is called recursively, follows from the induction hypothesis, as it is easy to see that the degree of C_3 in z is smaller than $\deg(F, z)$. Consequently, the **while** loop terminates, and the termination of the algorithm follows.

(Correctness) For the algorithm pgcd, if the input triangular set is simple set $\mathcal{S}$, then we can show that $\mathrm{sat}(\mathcal{S}_i)$ is also radical for any $i = 1, \ldots, s$ as follows. Let $\mathfrak{q}$ be any primary component of $\mathrm{sat}(\mathcal{S}_i)$ and $\mathfrak{p} = \sqrt{\mathfrak{q}}$. One knows from (b) that $\mathfrak{p}$ is a prime component of $\mathrm{sat}(\mathcal{S})$. Let $\mathcal{T} = \mathbb{K}[\boldsymbol{x}] \setminus \mathfrak{p}$. It is easy to verify that $\mathcal{T}$ intersects with every prime component of $\mathrm{sat}(\mathcal{S})$ other than $\mathfrak{p}$. By (a), $\mathrm{sat}(\mathcal{S}) \subseteq \mathrm{sat}(\mathcal{S}_i)$. Performing localization at $\mathcal{T}$ on this formula and then contracting back, we get $\mathfrak{p} \subseteq \mathfrak{q}$. Hence $\mathfrak{q} = \mathfrak{p}$, i.e. every primary component of $\mathrm{sat}(\mathcal{S}_i)$ is prime. Hence $\mathrm{sat}(\mathcal{S}_i)$ is radical. Again by Theorem 5.1.3, $\mathcal{S}_i$ is a simple set in $\mathbb{K}[\boldsymbol{x}]$. Furthermore, the ideal relation $\mathrm{sat}(\mathcal{S}) = \mathrm{sat}(\mathcal{S}_1) \cap \cdots \cap \mathrm{sat}(\mathcal{S}_s)$ is obtained, and thus Condition (a) is proved, combined with the irredundant property of the ideal decomposition by pgcd.

For any fixed path of one of the trees, we denote the node of depth i in the path by $[B(i), C(i), \mathcal{C}(i), \mathbb{P}(i), i]$, where $i \leq s$, the length of the path. For any associated prime $\mathfrak{p}$ of $\mathrm{sat}(\mathcal{C}(s))$ in the leaf node, $\overline{F}^{\mathfrak{p}}$ is a univariate polynomial over the field $\mathbb{F}_q[\boldsymbol{x}]/\mathfrak{p}$. Thus we can assume that $\overline{F}^{\mathfrak{p}} = Q \prod_{i=1}^{s-1} P_i^i$ is the decomposition of $\overline{F}^{\mathfrak{p}}$ as in Proposition 3.2.3. By the properties of pgcd, it is easy to check that $\overline{B(i)}^{\mathfrak{p}} \sim P_i P_{i+1} \cdots P_{s-1}$ and $\overline{C(i)}^{\mathfrak{p}} \sim Q P_{i+1} P_{i+2}^2 \cdots P_{s-1}^{s-i-1}$. In particular, $\overline{B(s)}^{\mathfrak{p}} \sim 1$ and $\overline{C(s)}^{\mathfrak{p}} \sim Q$. Hence $C(s)$ can be written in the form $\sum_i c_i \, z^{pi}$, where $c_i \in \mathbb{F}_q[\boldsymbol{x}]/\mathrm{sat}(\mathcal{C}(s))$. In the next step, if $\deg(C_1, z) = 0$, then the squarefree decomposition of F is obtained. Otherwise, lines 14.17–14.20 are executed to compute C_3, the pth root of $C(s)$ (i.e. C_1), and then the squarefree decomposition of C_3. The extraction of pth roots of polynomials in $(\mathbb{F}_q[\boldsymbol{x}]/\mathrm{sat}(\mathcal{S}))[z]$, where $\mathcal{S}$ is a zero-dimensional simple set in $\mathbb{F}_q[\boldsymbol{x}]$, will be discussed in the next subsection. Hence condition (b) of Definition 5.2.1 follows clearly from

the above analysis (analogous to the correctness proof of Algorithm 8). □

5.2.3 Extracting pth roots

Now we discuss the remaining issue in Algorithm 14: extracting the pth root of a polynomial $\sum_i c_i z^{pi} \in (\mathbb{F}_q[\boldsymbol{x}]/\operatorname{sat}(\mathcal{S}))[z]$, where $\mathcal{S}$ is a zero-dimensional simple set. This issue can be reduced to the extraction of the pth roots of the elements c_i in $\mathbb{F}_q[\boldsymbol{x}]/\operatorname{sat}(\mathcal{S})$. We already exploited the structure of $\mathbb{F}_q[\boldsymbol{x}]/\operatorname{sat}(\mathcal{S})$, which is isomorphic to a product of perfect fields of characteristic $p > 0$, in Section 5.1.

Proposition 5.2.4. *Let $\mathcal{S}$ be a zero-dimensional simple set in $\mathbb{F}_q[\boldsymbol{x}]$. Then for $i = 1, \ldots, n$, there exists a unique pth root for every element in $\mathbb{F}_q[\boldsymbol{x}_i]/\operatorname{sat}_i(\mathcal{S})$.*

Proof. The existence and uniqueness of the pth roots of elements in $\mathbb{F}_q[\boldsymbol{x}_i]/\operatorname{sat}_i(\mathcal{S})$ follow immediately from the fact that $\mathbb{F}_q[\boldsymbol{x}_i]/\operatorname{sat}_i(\mathcal{S})$ is isomorphic to a product of perfect fields. □

For a positive-dimensional simple set $\mathcal{S}$, $\tilde{\mathbb{F}}_q = \mathbb{F}_q(\boldsymbol{u})$ is not perfect, so the extraction of pth roots in $\tilde{\mathbb{F}}_q[\boldsymbol{y}_i]/\operatorname{sat}_i(\mathcal{S})_{\tilde{\mathbb{F}}_q}$ may be infeasible. Consider for example $\mathcal{T} = [y^3 - u] \subseteq \mathbb{F}_3[u, y]$. The 3rd root of u does not exist in $\mathbb{F}_3(u)[y]/\langle y^3 - u\rangle_{\mathbb{F}_3(u)}$. We need some extra technique other than the one presented here to handle the positive-dimensional case. We will study this problem in Section 5.3.

For any perfect field of characteristic $p > 0$, one can extract the pth roots of its elements by solving some linear equations (see, e.g., [68]). In view of the product structure of $\mathbb{F}_q[\boldsymbol{x}]/\operatorname{sat}(\mathcal{S})$, an obvious way for extracting the pth root of an element $F \in \mathbb{F}_q[\boldsymbol{x}]/\operatorname{sat}(\mathcal{S})$ is to compute the pth root of $\pi_j(F)$ over all the branches $\mathbb{F}_q[\boldsymbol{x}]/\mathfrak{p}_j$ and then lift them back, where $\mathfrak{p}_j$ $(j = 1, \ldots, s)$ are the associated primes of $\operatorname{sat}(\mathcal{S})$. The drawback of this method is that it needs to split the field product $\mathbb{F}_q[\boldsymbol{x}]/\operatorname{sat}(\mathcal{S})$ completely. As prime decomposition of $\operatorname{sat}(\mathcal{S})$ may involve the computation of Gröbner bases or irreducible triangular sets, one can imagine the ineffectiveness of this method.

In what follows, we propose another method for pth root extraction. The following two propositions serve as the basis of our method.

Proposition 5.2.5. *Let $\mathcal{S} = [S_1, \ldots, S_r] \subseteq \mathbb{K}[\boldsymbol{x}]$ be a simple set. Then, for any $i = 1, \ldots, r$, $\tilde{\mathbb{K}}[\boldsymbol{y}_i]/\operatorname{sat}_i(\mathcal{S})_{\tilde{\mathbb{K}}}$ is a $(\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\operatorname{sat}_{i-1}(\mathcal{S})_{\tilde{\mathbb{K}}})$-module and $1, y_i, \ldots, y_i^{d-1}$ form a free basis (of this module), where $d = \deg(S_i, y_i)$.*

Proof. First we have

$$\tilde{\mathbb{K}}[\boldsymbol{y}_i]/\operatorname{sat}_i(\mathcal{S})_{\tilde{\mathbb{K}}} \cong (\tilde{\mathbb{K}}[\boldsymbol{y}_i]/\operatorname{sat}_{i-1}(\mathcal{S})_{\tilde{\mathbb{K}}})/(\operatorname{sat}_i(\mathcal{S})_{\tilde{\mathbb{K}}}/\operatorname{sat}_{i-1}(\mathcal{S})_{\tilde{\mathbb{K}}}).$$

Furthermore, $\tilde{\mathbb{K}}[\boldsymbol{y}_i]/\operatorname{sat}_{i-1}(\mathcal{S})_{\tilde{\mathbb{K}}} = (\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\operatorname{sat}_{i-1}(\mathcal{S})_{\tilde{\mathbb{K}}})[y_i]$ and $\operatorname{sat}_i(\mathcal{S})_{\tilde{\mathbb{K}}}/\operatorname{sat}_{i-1}(\mathcal{S})_{\tilde{\mathbb{K}}} = \langle \overline{S}_i \rangle$, which is an ideal in $(\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\operatorname{sat}_{i-1}(\mathcal{S})_{\tilde{\mathbb{K}}})[y_i]$. It follows that $\tilde{\mathbb{K}}[\boldsymbol{y}_i]/\operatorname{sat}_i(\mathcal{S})_{\tilde{\mathbb{K}}}$ is a $(\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\operatorname{sat}_{i-1}(\mathcal{S})_{\tilde{\mathbb{K}}})$-module. Let $\overline{F} \in (\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\operatorname{sat}_{i-1}(\mathcal{S})_{\tilde{\mathbb{K}}})[y_i]$. Since $\operatorname{lc}(\overline{T}_i)$ is regular and thus invertible in $\tilde{\mathbb{K}}[\boldsymbol{y}_{i-1}]/\operatorname{sat}_{i-1}(\mathcal{S})_{\tilde{\mathbb{K}}}$, we can divide $\overline{F}$ by $\overline{S}_i$. The remainder is a linear combination of $1, y_i, \ldots, y_i^{d-1}$, which is equal to $\overline{F}$ in $\tilde{\mathbb{K}}[\boldsymbol{y}_i]/\operatorname{sat}_i(\mathcal{S})_{\tilde{\mathbb{K}}}$. It is easy to verify the linear independence of $1, y_i, \ldots, y_i^{d-1}$. □

The elements $1, y_i, \ldots, y_i^{d-1}$ in the above proposition are called the *standard basis* of $\tilde{\mathbb{K}}[\boldsymbol{y}_i]/\operatorname{sat}_i(\mathcal{S})_{\tilde{\mathbb{K}}}$.

Proposition 5.2.6. *Let $\mathcal{R}$ be a ring isomorphic to a product of fields, M be an $n \times n$ matrix over $\mathcal{R}$, and $\boldsymbol{b} \in \mathcal{R}^n$. Then the set of linear equations*

$$M\boldsymbol{x} = \boldsymbol{b} \tag{5.2}$$

has a unique solution if and only if $\det(M)$ is regular in $\mathcal{R}$. If the equivalent conditions are satisfied, then the unique solution is $\boldsymbol{a} = (a_1, \ldots, a_s)$, where $a_i = \det(M_i) \cdot \det(M)^{-1}$ and M_i is the matrix obtained by replacing the ith column of M with $\boldsymbol{b}$.

Proof. Suppose that $\mathcal{R} \cong \mathbb{K}_1 \times \cdots \times \mathbb{K}_s$, where $\mathbb{K}_i$ is a field for $i = 1, \ldots, s$. We use π_i to denote the projection of $\mathcal{R}$ to $\mathbb{K}_i$. It induces two maps respectively from matrices and vectors over $\mathcal{R}$ to those over $\mathbb{K}_i$, which are also denoted by π_i. If $\boldsymbol{a} = (a_1, \ldots, a_n)$ is a solution of (5.2), then the following sets of equations may be obtained by projection:

$$\pi_i(M)\,\pi_i(\boldsymbol{a}) = \pi_i(\boldsymbol{b}), \quad i = 1, \ldots, s. \tag{5.3}$$

Thus solving (5.2) for $\boldsymbol{x}$ is equivalent to finding $\pi_i(\boldsymbol{a})$ satisfying (5.3) for all $i = 1, \ldots, s$. According to the Cramer's rule, each set of equations in (5.3) has a unique

solution if and only if $\det(\pi_i(M)) = \pi_i(\det(M)) \neq 0$; if the equivalent conditions are satisfied, then $\pi_i(a_j) = \det(\pi_i(M_j)) \cdot \det(\pi_i(M))^{-1} = \pi_i(\det(M_j) \cdot \det(M)^{-1})$, where M_i is the matrix obtained by replacing the ith column of M with $\boldsymbol{b}$.

Hence (5.2) has a unique solution if and only if $\det(M)$ is regular in $\mathcal{R}$. If the equivalent conditions are satisfied, then one can find the unique solution $\boldsymbol{a} = (a_1, \ldots, a_s)$ with $a_i = \det(M_i) \cdot \det(M)^{-1}$. □

In the following parts of this section, the simple set $\mathcal{S} = [S_1, \ldots, S_n] \subseteq \mathbb{F}_q[\boldsymbol{x}]$ is restricted to be zero-dimensional. Let $F \in \mathbb{F}_q[\boldsymbol{x}]$, $\mathrm{lv}(F) = x_i$, and $\deg(T_i, x_i) = d$. We want to construct a polynomial $G = a_0 + a_1 x_i + \cdots + a_{d-1} x_i^{d-1}$ with $a_j \in \mathbb{F}_q[\boldsymbol{x}_{i-1}]$ such that $G^p = F$ modulo $\mathrm{sat}_i(\mathcal{S})$.

Suppose that the pseudo-division formula of $G^p - F$ w.r.t. S_i is $C(G^p - F) = QS_i + R$, where C is some power of $\mathrm{ini}(S_i)$ and $R = \mathrm{prem}(G^p - F, S_i)$. The pseudo-remainder R can be written as $R = (f_{d-1} x_i^{d-1} + \cdots + f_0) + (b_{d-1} x_i^{d-1} + \cdots + b_0)$, where each f_j is a linear combination of $a_0^p, \ldots, a_{d-1}^p$ with coefficient in $\mathbb{F}_q[\boldsymbol{x}_{i-1}]$ and each b_j is an element in $\mathbb{F}_q[\boldsymbol{x}_{i-1}]$ for $j = 0, \ldots, d-1$. Noting that C is regular modulo $\mathrm{sat}_i(\mathcal{S})$, one can prove that $G^p = F$ modulo $\mathrm{sat}_i(\mathcal{S})$ is equivalent to

$$f_{d-1} x_i^{d-1} + \cdots + f_0 = -(b_{d-1} x_i^{d-1} + \cdots + b_0) \ \text{ modulo } \ \mathrm{sat}_i(\mathcal{S}).$$

Comparing the coefficients of x_i^j $(j = 0, \ldots, d-1)$ in this equality, we obtain the following set of equations modulo $\mathrm{sat}_{i-1}(\mathcal{S})$:

$$\begin{aligned} f_0(a_0^p, \ldots, a_{d-1}^p) &= -b_0, \\ f_1(a_0^p, \ldots, a_{d-1}^p) &= -b_1, \\ &\vdots \\ f_{d-1}(a_0^p, \ldots, a_{d-1}^p) &= -b_{d-1}. \end{aligned} \tag{5.4}$$

As the existence and uniqueness of the pth roots of elements in $\mathbb{F}_q[\boldsymbol{x}_i]/\mathrm{sat}_i(\mathcal{S})$ have been proven, the set of linear equations (5.4) has a unique solution; hence we are able to solve it for $a_0^p, \ldots, a_{d-1}^p$ by Proposition 5.2.6. After a_j^p is obtained, we are in the position to compute the pth root a_j of a_j^p in $\mathbb{F}_q[\boldsymbol{x}_{i-1}]/\mathrm{sat}_{i-1}(\mathcal{S})$ for $j = 0, \ldots, d-1$. Repeating the above process will lead to pth root extraction in $\mathbb{F}_q$ in the end, which is computable. It should be noted that when using Proposition 5.2.6 to solve (5.4),

one needs to obtain the inverse of an element in $\mathbb{F}_q[\boldsymbol{x}_{i-1}]/\operatorname{sat}_{i-1}(\mathcal{S})$, which can be computed by the algorithm QuasiRecip described in [113].

The whole process is illustrated by the following example.

Example 5.2.1. Consider the simple set $\mathcal{S} = [S_1, S_2] = [x^2 + 2\,x + 2, x\,y^2 + y + 1] \subseteq \mathbb{F}_5[x, y]$. We want to compute the 5th root of $F = y + 4\,x$ in $\mathbb{F}_5[x, y]/\operatorname{sat}(\mathcal{S})$. Suppose that $G = a_1\,y + a_0$ is the 5th root of F. One has the pseudo-division formula of $G^5 - F$ w.r.t. S_2:

$$x^4(G^5 - F) = \operatorname{pquo}(G^5, S_2) \cdot S_2 + (x^2 a_1^5 + 2\,x\,a_1^5 + a_1^5)\,y + (x^4 a_0^5 + 3\,x\,a_1^5 + a_1^5) - (x^4\,y + 4\,x^5).$$

And thus in $\mathbb{F}_5[x, y]/\operatorname{sat}(\mathcal{S})$, we have $(x^2 a_1^5 + 2\,x\,a_1^5 + a_1^5)\,y + (x^4 a_0^5 + 3\,x\,a_1^5 + a_1^5) - (x^4\,y + 4\,x^5) = 0$. Equating the coefficients of y^i $(i = 1, 0)$ to 0, we obtain a set of linear equations in the matrix form

$$\begin{pmatrix} x^4 & 3\,x + 1 \\ 0 & x^2 + 2\,x + 1 \end{pmatrix} \begin{pmatrix} a_0^5 \\ a_1^5 \end{pmatrix} = \begin{pmatrix} 4\,x^5 \\ x^4 \end{pmatrix} \tag{5.5}$$

Denote by M the coefficient matrix of (5.5) and by M_i the matrix obtained by replacing the $(i+1)$th column of M with $\begin{bmatrix} 4\,x^5 & x^4 \end{bmatrix}^T$ for $i = 0, 1$. By Proposition 5.2.6, $a_i^5 = \det(M_i) \cdot \det(M)^{-1}$ is the solution of (5.5). Using the algorithm QuasiRecip in [113], we find that the inverse of $\det(M) = x^4\,(x^2 + 2\,x + 1)$ is 4. Hence the solution of (5.5) is $a_0^5 = x^7 + 2\,x^6 + 4\,x^5 + x^4$ and $a_1^5 = 4\,x^8$. Recursively extracting the 5th root of a_0^5 and a_1^5 in the same way, we get $G = 4\,y + 2\,x + 1$ in the end.

5.2.4 Algorithm for simple decomposition: zero-dimensional case

In this section, we present our algorithms to decompose zero-dimensional polynomial sets into simple sets over finite fields.

Let $\mathcal{F}$ be a zero-dimensional polynomial set in $\mathbb{F}_q[\boldsymbol{x}]$. The following algorithm computes a finite number of simple sets $\mathcal{S}_1, \ldots, \mathcal{S}_r$ such that $\sqrt{\langle \mathcal{F} \rangle} = \bigcap_{i=1}^{r} \langle \mathcal{S}_i \rangle$.

Algorithm 15: Simple decomposition zero-dim $\mathbb{S} := \mathsf{SimDecZero}(\mathcal{F})$

Input: $\mathcal{F}$ — a zero-dimensional polynomial set in $\mathbb{F}_q[\boldsymbol{x}]$.

Output: $\mathbb{S}$ — a finite set of simple sets in $\mathbb{F}_q[\boldsymbol{x}]$ such that $\sqrt{\langle\mathcal{F}\rangle} = \bigcap_{\mathcal{S}\in\mathbb{S}}\langle\mathcal{S}\rangle$.

```
15.1  S := ∅;
15.2  for T ∈ RegDec(F) do
15.3      D := {(T, ∅)}; S_T := ∅;
15.4      while D ≠ ∅ do
15.5          (A, S) := pop(D);
15.6          if A = ∅ then
15.7              S_T := S_T ∪ {S};
15.8          else
15.9              A := the polynomial in A with smallest leading variable;
15.10             for ({[C_1, c_1], ..., [C_s, c_s]}, Q) ∈ sqfZero(A, S) do
15.11                 D := D ∪ {(A \ {A}, Q ∪ {C_1 ··· C_s})};
15.12             end
15.13         end
15.14     end
15.15     S := S ∪ S_T;
15.16 end
```

Proof. It is easy to verify the termination of the algorithm. We prove the correctness as follows.

For any zero-dimensional regular set $\mathcal{T}$, $\mathrm{sat}(\mathcal{T}) = \langle\mathcal{T}\rangle$. By Specification 6, $\sqrt{\langle\mathcal{F}\rangle} = \bigcap_{\mathcal{T}\in\mathsf{RegDec}(\mathcal{F})}\sqrt{\langle\mathcal{T}\rangle}$. By Algorithm 14, one can easily verify that for any $(\mathcal{A},\mathcal{S}) \in \mathbb{D}$, $\mathcal{S}$ is a simple set. Hence each element in $\mathbb{S}$ is a simple set.

To prove the ideal relation $\sqrt{\langle\mathcal{F}\rangle} = \bigcap_{\mathcal{S}\in\mathbb{S}}\langle\mathcal{S}\rangle$, we only need to prove that for each $\mathcal{T} \in \mathsf{RegDec}(\mathcal{F})$, $\sqrt{\langle\mathcal{T}\rangle} = \bigcap_{\mathcal{S}\in\mathbb{S}_\mathcal{T}}\langle\mathcal{S}\rangle$ holds at the end of the corresponding **while** loop for $\mathcal{T}$. For Lines 15.10–15.12, the ideal relation

$$\sqrt{\langle\mathcal{A}\rangle + \langle\mathcal{S}\rangle} = \bigcap_{(\{[C_1,c_1],\ldots,[C_s,c_s]\},\mathcal{Q})\in\mathsf{sqfZero}(A,\mathcal{S})} \sqrt{\langle\mathcal{A}\rangle + \langle\mathcal{Q}\rangle}$$

holds. For each $(\{[C_1, c_1], \ldots, [C_s, c_s]\}, \mathcal{Q}) \in \mathsf{sqfZero}(A, \mathcal{S})$, we have

$$\begin{aligned}\sqrt{\langle \mathcal{A} \rangle + \langle \mathcal{Q} \rangle} &= \sqrt{\langle \mathcal{A} \setminus \{A\} \rangle + \langle \mathcal{Q} \cup \{A\} \rangle} \\ &= \sqrt{\langle \mathcal{A} \setminus \{A\} \rangle + \sqrt{\langle \mathcal{Q} \cup \{A\} \rangle}} \\ &= \sqrt{\langle \mathcal{A} \setminus \{A\} \rangle + \langle \mathcal{Q} \cup \{C_1 \cdots C_s\} \rangle}.\end{aligned}$$

Hence the following invariant of the corresponding **while** loop for $\mathcal{T}$ follows:

$$\sqrt{\langle \mathcal{T} \rangle} = \bigcap_{(\mathcal{A}, \mathcal{S}) \in \mathbb{D}} \sqrt{\langle \mathcal{S} \cup \mathcal{A} \rangle} \cap \bigcap_{\mathcal{S} \in \mathbb{S}_{\mathcal{T}}} \langle \mathcal{S} \rangle. \qquad \square$$

Next we present prejudging criteria for excluding some cases in which complete squarefree decomposition is unnecessary and some variants of the algorithm SimDecZero.

In the case of univariate polynomials, by Proposition 3.2.3 a polynomial $F \in \mathbb{F}_q[x]$ can be written in the form $F = Q \prod_i P_i^i$ with $Q' = 0$. If Q is a constant, then the squarefree part of Q can be obtained just by computing $F/\gcd(F, F')$; otherwise, we need squarefree decomposition as done in Algorithm 8. In other words, squarefree decomposition in the cases when Q is a constant may be avoided by identifying such cases. The key observation is as follows.

If Q is not a constant, then Q is a pth power of some nonconstant polynomial. In this case, the following conditions must be satisfied:

- $\deg(F) \geq p$;
- if $F^{(s)}$ is the last nonzero polynomial in the derivative sequence

  $$F, F', F^{(2)}, \ldots, F^{(s)}, 0, \ldots,$$

  then $p \mid \deg(F^{(s)})$.

Let $\mathcal{S}$ be a simple set in $\mathbb{F}_q[\boldsymbol{x}]$ and $F \in \mathbb{F}_q[\boldsymbol{x}][z]$. These conditions can be generalized to the case of generalized squarefree decomposition:

- $\deg(F, z) \geq p$;
- if $\partial^s F/\partial z^s$ is the last regular polynomial modulo $\mathrm{sat}(\mathcal{S})$ in the derivative sequence $F, \partial F/\partial z, \partial^2 F/\partial z^2, \ldots, \partial^s F/\partial z^s, \ldots$, then $p \mid \deg(\partial^s F/\partial z^s, z)$.

The above conditions can be used to identify some cases in which the technique for computing the squarefree part in still works. However, for the second condition it is necessary to determine whether a polynomial is regular, which may be quite time-consuming as shown in Section 5.4.1. The modified version of SimDecZero with the prejudging criteria incorporated is named as SimDecZeroPJ.

What we have actually computed is the complete squarefree decomposition, while only the squarefree part is needed. One can choose to split the squarefree part into factors. The splitting may lead to more branches, but polynomials in each branch are simpler. We call the new algorithm with this splitting strategy the *strong simple decomposition algorithm*, denoted by SSimDecZero, which can be easily obtained by replacing Line 15.11 of Algorithm 15 with the following statement

$$\text{“}\mathbb{D} := \mathbb{D} \cup \{(\mathcal{A} \setminus \{A\}, \mathcal{Q} \cup \{C_i\}) : i = 1, \ldots, s\};\text{”}.$$

The output of SSimDecZero has the same properties as that of SimDecZero, and the proof of termination and correctness is similar.

For instance, consider the regular set $\mathcal{T} = [(x+1)^4(x^3+2\,x+1), y^3+x+2]$ in the polynomial ring $\mathbb{F}_3[x,y]$ with $x < y$. SSimDecZero($\mathcal{T}$) yields two simple sets $[x+1, y+1]$ and $[x^3+2\,x+1, y+x]$, while SimDecZero($\mathcal{T}$) returns $[x^4+x^3+2\,x^2+1, y+2\,x^3+2\,x+2]$.

The following example illustrates the entire process of our main algorithms.

Example 5.2.2. Consider the polynomial set $\mathcal{F} = \{F_1, F_2\} \subseteq \mathbb{F}_3[x,y]$, where

$$F_1 = (y^2+y+2\,x^2+2)(2\,xy+y+2\,x^2+2)(x^6y^6+2\,x^9y^3+2\,x^3y^3+x^{12}+2\,x^6+1),$$
$$F_2 = 2\,y^6+y^3+2\,x^6+1.$$

Order the variables as $x < y$. By the algorithm RegDec, $\mathcal{F}$ is decomposed into four regular sets

$$\mathcal{T}_1 = [x+1, y+2], \quad \mathcal{T}_2 = [x+2, y+2], \quad \mathcal{T}_3 = [x^2+1, y+1], \quad \mathcal{T}_4 = [T_1, T_2],$$

where

$$T_1 = x^3+2\,x+1, \quad T_2 = (x^2+2)\,y^4 + (x^2+2\,x)\,y^3 + (2\,x^2+x+1)\,y + x^2+2.$$

It is easy to check that $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ are all simple sets, but $\mathcal{T}_4$ is not. Thus further computation is needed to turn $\mathcal{T}_4$ into simple sets. In SimDecZero, $\mathcal{T}_4$ is converted to

$$\mathcal{S}_4 = [T_1, (x^4 + x^2 + x)\, y^2 + (2\, x^4 + 2\, x^3 + 2\, x^2 + 2)\, y + 2\, x^3 + 2\, x + 2],$$

which is a simple set. The output of SimDecZeroPJ is the same. To illustrate the prejudging criteria in SimDecZeroPJ, first consider T_1. Note that $\deg(T_1, x) = 3$ and the derivative sequence of T_1 is $T_1, 2, 0$. Thus the second prejudging criterion ensures that complete squarefree decomposition of T_1 is unnecessary and one can use $\gcd(T_1, T_1')/T_1$ only to obtain the squarefree part of T_1. For T_2, $\deg(T_2, y) > 3$ and its derivative sequence $T_2, 2\, y^3 + x^2 y^2 + 2\, x^2 + x + 1, 0$ satisfies the second prejudging criterion. Hence squarefree decomposition is still needed.

The algorithm SSimDecZero produces a finer decomposition, turning $\mathcal{T}_4$ into two simple sets

$$\mathcal{S}_{41} = [T_1, (x^2 + x + 2)\, y + x + 2], \quad \mathcal{S}_{42} = [T_1, (x^2 + 2\, x)\, y + 2\, x^2 + 2\, x + 1].$$

Consequently, two kinds of simple decomposition of $\mathcal{F}$ are obtained such that

$$\sqrt{\langle \mathcal{F} \rangle} = \bigcap_{i=1}^{3} \langle \mathcal{T}_i \rangle \cap \langle \mathcal{S}_4 \rangle = \bigcap_{i=1}^{3} \langle \mathcal{T}_i \rangle \cap \langle \mathcal{S}_{41} \rangle \cap \langle \mathcal{S}_{42} \rangle.$$

5.3 Positive-dimensional polynomial sets

As already explained in Section 5.2.3, the technique for pth root extraction is no longer applicable to the positive-dimensional case. Therefore in this case we have to find an alternative way to design the algorithm for simple decomposition.

By adjoining the parameters of the regular set to the ground finite field, the problem of simple decomposition over the ground field in the positive-dimensional case is then reduced to that over a purely transcendental extension of the ground field in the zero-dimensional case. We propose a method to solve the latter problem with the critical step of computing the radical of a special ideal over the field extension.

Computation of radicals of positive-dimensional ideals over finite fields is a difficult problem itself, but fortunately it has already been studied in the literature. We investigate the three methods developed by Matsumoto [106], Kemper [83], and Fortuna, Gianni, and Trager (referred to as *FGT*) [62] for the computation of radicals

and, in particular, improve the FGT method (which is the most suitable) according to our specific setting to formulate a simplified and optimized algorithm.

5.3.1 Algorithm for simple decomposition: positive-dimensional case

We start our solving approach to the problem formulated at the end of Section 5.1 with the following observations.

Proposition 5.3.1. *Let $\mathcal{T} \subset \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}]$ be a normal triangular set. Then $\mathcal{T}$ is a Gröbner basis of $\langle \mathcal{T} \rangle_{\tilde{\mathbb{K}}}$ w.r.t. the lexicographical term order.*

Proof. For any two distinct polynomials in $\mathcal{T}$, viewed as elements in $\tilde{\mathbb{K}}[\boldsymbol{y}]$, their leading terms are powers of their respective leading variables and thus they are coprime. The conclusion follows from Proposition 4 in [35, Section 2.9]. □

Lemma 5.3.2 ([90]). *Let $\mathcal{T}$ be a regular set in $\mathbb{K}[\boldsymbol{u}][\boldsymbol{y}]$, and $P \in \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}]$. Then $P \in \mathrm{sat}(\mathcal{T})$ if and only if there exists a nonzero polynomial $L \in \mathbb{K}[\boldsymbol{u}]$ such that $LP \in \langle \mathcal{T} \rangle$.*

Proposition 5.3.3. *For any regular set $\mathcal{T} \subset \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}]$, $\mathrm{sat}(\mathcal{T}) = \langle \mathcal{T} \rangle_{\tilde{\mathbb{K}}} \cap \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}]$.*

Proof. ($\subset$) By Proposition 2.3.3, $\mathrm{sat}(\mathcal{T})_{\tilde{\mathbb{K}}} = \langle \mathcal{T} \rangle_{\tilde{\mathbb{K}}}$ and thus

$$\mathrm{sat}(\mathcal{T}) \subset \mathrm{sat}(\mathcal{T})_{\tilde{\mathbb{K}}} \cap \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}] = \langle \mathcal{T} \rangle_{\tilde{\mathbb{K}}} \cap \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}].$$

($\supset$) Let $\mathcal{T} = [T_1, \ldots, T_r]$. Then for any $P \in \langle \mathcal{T} \rangle_{\tilde{\mathbb{K}}} \cap \mathbb{K}[\boldsymbol{u}][\boldsymbol{y}]$, there exist $Q_1, \ldots, Q_r \in \tilde{\mathbb{K}}[\boldsymbol{y}]$ such that $P = \sum_{i=1}^{r} Q_i T_i$. Let L be the least common multiplier of the denominators of all the coefficients of $Q_1, \ldots, Q_r$. Then $L \in \mathbb{K}[\boldsymbol{u}] \setminus \{0\}$ and $LP \in \langle \mathcal{T} \rangle$. Hence $P \in \mathrm{sat}(\mathcal{T})$ by Lemma 5.3.2. □

Proposition 5.3.4. *Let $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{K}[\boldsymbol{x}]$ be a regular set with $\mathrm{lv}(T_i) = y_i$ for $i = 1, \ldots, r$. If $[T_1, \ldots, T_{r-1}]$ is a simple set, then*

$$\sqrt{\langle \mathcal{T} \rangle_{\tilde{\mathbb{K}}}} \cap \tilde{\mathbb{K}}[y_1, \ldots, y_{r-1}] = \langle T_1, \ldots, T_{r-1} \rangle_{\tilde{\mathbb{K}}}.$$

Proof. First we see, for instance, from [4, Exercise 1.13] that

$$\sqrt{\langle \mathcal{T} \rangle_{\tilde{\mathbb{K}}}} \cap \tilde{\mathbb{K}}[y_1, \ldots, y_{r-1}] = \sqrt{\langle \mathcal{T} \rangle_{\tilde{\mathbb{K}}}} \cap \sqrt{\tilde{\mathbb{K}}[y_1, \ldots, y_{r-1}]} = \sqrt{\langle \mathcal{T} \rangle_{\tilde{\mathbb{K}}} \cap \tilde{\mathbb{K}}[y_1, \ldots, y_{r-1}]}.$$

Let $\mathcal{N} = [N_1, \ldots, N_r]$ be a normal triangular set transformed from $\mathcal{T}$ such that $\mathrm{sat}(\mathcal{N}) = \mathrm{sat}(\mathcal{T})$. Then by Proposition 5.3.1, $\mathcal{N}$ is a Gröbner basis of $\langle T\rangle_{\tilde{\mathbb{K}}}$ w.r.t. the lexicographical term order. Thus

$$\langle \mathcal{T}\rangle_{\tilde{\mathbb{K}}} \cap \tilde{\mathbb{K}}[y_1, \ldots, y_{r-1}] = \langle N_1, \ldots, N_{r-1}\rangle_{\tilde{\mathbb{K}}} = \langle T_1, \ldots, T_{r-1}\rangle_{\tilde{\mathbb{K}}},$$

where the last equality follows from Proposition 2.3.4(a) and the fact that $\mathrm{sat}(\mathcal{N}) = \mathrm{sat}(\mathcal{T})$. □

Let $\mathcal{T} = [T_1, \ldots, T_r] \subset \mathbb{F}_q[\boldsymbol{x}]$ be a regular set with $\mathrm{lv}(T_i) = y_i$ and $[T_1, \ldots, T_{r-1}]$ already a simple set. According to Proposition 5.3.4, we assume that $\sqrt{\langle \mathcal{T}\rangle_{\tilde{\mathbb{F}}_q}}$ can be written as

$$\sqrt{\langle \mathcal{T}\rangle_{\tilde{\mathbb{F}}_q}} = \langle T_1, \ldots, T_{r-1}, F_1, \ldots, F_k\rangle_{\tilde{\mathbb{F}}_q},$$

where $F_j \in \tilde{\mathbb{F}}_q[\boldsymbol{y}]$ and $\mathrm{lv}(F_j) = y_s$ for $j = 1, \ldots, k$. We turn F_j into a polynomial in $\mathbb{F}_q[\boldsymbol{u}][\boldsymbol{y}]$ by multiplying some polynomial in $\mathbb{F}_q[\boldsymbol{u}]$ as usual and denote the resulting polynomial still by F_j.

Now take

$$\mathbb{G} := \mathsf{pgcd}(\{F_1, \ldots, F_k\}, [T_1, \ldots, T_{r-1}]).$$

Then for any $(P, \mathcal{A}) \in \mathbb{G}$ and associate prime $\mathfrak{p}$ of $\mathrm{sat}(\mathcal{A})_{\tilde{\mathbb{F}}_q}$, we have $\overline{P}^{\mathfrak{p}} = \gcd(\overline{F}_1^{\mathfrak{p}}, \ldots, \overline{F}_k^{\mathfrak{p}})$, namely $\langle \overline{P}^{\mathfrak{p}}\rangle = \langle \overline{F}_1^{\mathfrak{p}}, \ldots, \overline{F}_k^{\mathfrak{p}}\rangle$. It follows that $\langle \overline{P}\rangle = \langle \overline{F}_1, \ldots, \overline{F}_k\rangle$ and

$$\sqrt{\langle \mathcal{T}\rangle_{\tilde{\mathbb{F}}_q}} = \langle T_1, \ldots, T_{r-1}, F_1, \ldots, F_k\rangle_{\tilde{\mathbb{F}}_q} = \bigcap_{(P,\mathcal{A})\in\mathbb{G}} \langle \mathcal{A} \cup \{P\}\rangle_{\tilde{\mathbb{F}}_q}.$$

Contracting the above equation back to $\mathbb{F}_q[\boldsymbol{u}][\boldsymbol{y}]$ (i.e., taking intersection with $\mathbb{F}_q[\boldsymbol{u}][\boldsymbol{y}]$) and by Proposition 5.3.4, we have

$$\sqrt{\mathrm{sat}(\mathcal{T})} = \bigcap_{(P,\mathcal{A})\in\mathbb{G}} \mathrm{sat}(\mathcal{A} \cup \{P\}). \tag{5.6}$$

Lemma 5.3.5 ([98, Lemma 3.2]). *Let $\mathcal{T} = [T_1, \ldots, T_r]$ be any regular set in $\mathbb{K}[\boldsymbol{x}]$ and $F \in \mathbb{K}[\boldsymbol{x}]$ with $\mathrm{lv}(F) = y_i$ $(1 \le i \le r)$. If $\overline{F} \in \langle \overline{T}_i\rangle$ and $\mathrm{sat}_{i-1}(\mathcal{T})_{\tilde{\mathbb{K}}}$ is radical, then $F \in \mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}}$.*

Proposition 5.3.6. *For every $(P, \mathcal{A}) \in \mathbb{G}$ in the decomposition (5.6), $\mathcal{A}\cup\{P\}$, when ordered properly, is a simple set.*

Proof. Since $[T_1, \ldots, T_{r-1}]$ is a simple set, $\mathrm{sat}([T_1, \ldots, T_{r-1}])$ is radical by Theorem 5.1.3. Thus it follows from Specification 6 that $\mathrm{sat}(\mathcal{A})$ is radical for every $(P, \mathcal{A}) \in \mathbb{G}$. Hence $\langle \mathcal{A} \rangle_{\tilde{\mathbb{F}}_q} = \mathrm{sat}(\mathcal{A})_{\tilde{\mathbb{F}}_q}$ is also radical. From the radicality of $\langle T_1, \ldots, T_{r-1}, F_1, \ldots, F_k \rangle_{\tilde{\mathbb{F}}_q}$, we know that $\langle \overline{P} \rangle = \langle \overline{F}_1, \ldots, \overline{F}_k \rangle$ is radical, where $\overline{F}$ denotes the image of F in $(\tilde{\mathbb{F}}_q[\boldsymbol{y}_{s-1}]/\mathrm{sat}(\mathcal{A})_{\tilde{\mathbb{F}}_q})[y_s]$.

We complete the proof by showing that $\langle \mathcal{A} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q}$ is radical. Suppose that $F^n \in \langle \mathcal{A} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q}$. Then $\overline{F}^n \in \langle \overline{P} \rangle$ and hence $\overline{F} \in \langle \overline{P} \rangle$, because $\langle \overline{P} \rangle$ is radical. By Lemma 5.3.5, $F \in \langle \mathcal{A} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q}$. This implies that $\langle \mathcal{A} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q}$ is radical. □

Now we are ready to formulate the method described above as Algorithm 16.

Algorithm 16: Simple decomposition Positive-dim $\mathbb{S} = \mathsf{SimDecPos}(\mathcal{T})$

Input: $\mathcal{T}$ — a regular set in $\mathbb{F}_q[\boldsymbol{x}]$.

Output: $\mathbb{S}$ — a finite set of simple sets in $\mathbb{F}_q[\boldsymbol{x}]$, such that $\sqrt{\mathrm{sat}(\mathcal{T})} = \bigcap_{\mathcal{S} \in \mathbb{S}} \mathrm{sat}(\mathcal{S})$.

16.1 $\mathbb{S} := \emptyset$; $\mathbb{D} := \{(\mathcal{T}, \emptyset)\}$;
16.2 **while** $\mathbb{D} \neq \emptyset$ **do**
16.3 $(\mathcal{A}, \mathcal{S}) := \mathsf{pop}(\mathbb{D})$;
16.4 **if** $\mathcal{A} = \emptyset$ **then**
16.5 $\mathbb{S} := \mathbb{S} \cup \{\mathcal{S}\}$;
16.6 **else**
16.7 $T :=$ polynomial in $\mathcal{A}$ with smallest leading variable y_t;
16.8 $\mathcal{F} :=$ LEX Gröbner basis of $\sqrt{\langle \mathcal{S} \cup \{T\} \rangle_{\tilde{\mathbb{F}}_q}}$;
16.9 $\mathcal{F}' := \{F \in \mathcal{F} : \mathrm{lv}(F) = y_t\}$;
16.10 $\mathbb{G} := \mathsf{pgcd}(\mathcal{F}', \mathcal{S})$;
16.11 $\mathbb{D} := \mathbb{D} \cup \{(\mathcal{A} \setminus \{T\}, \mathcal{Q} \cup \{F\}) : (F, \mathcal{Q}) \in \mathbb{G}\}$;
16.12 **end**
16.13 **end**
16.14 **return** $\mathbb{S}$

To use Algorithm 16, one needs to compute the Gröbner bases of the radicals of zero-dimensional ideals over a purely transcendental extension of $\mathbb{F}_q$. This is a major computational step and will be discussed in the next subsection.

Example 5.3.1. We illustrate how Algorithm 16 works by a simple example: $\mathcal{T} =$

$[T_1, T_2] \subseteq \mathbb{F}_3[s,t,x,y]$ with $s < t < x < y$, where

$$T_1 = (x^9 - s)(x^3 - t), \quad T_2 = y^3 - t.$$

It is easy to verify that $\mathcal{T}$ is a regular set.

In the first iteration of Algorithm 16, $(\mathcal{A}, \mathcal{S}) = (\mathcal{T}, \emptyset)$ and T_1 is chosen as the polynomial with smallest leading variable. The result of Line 16.10 is $\mathbb{G} = \{(T_1, \emptyset)\}$ and thus, after this iteration, $\mathbb{D} = \{([T_2], [T_1])\}$.

In the second iteration, we have $(\mathcal{A}, \mathcal{S}) = ([T_2], [T_1])$. The polynomial in Line 16.7 is T_2 and the Gröbner basis of $\sqrt{\langle T_1, T_2\rangle_{\tilde{\mathbb{F}}_q}}$ computed is $\mathcal{F} = \{T_1, T_2, (y-x)(x^9-s)\}$. Therefore we have

$$\mathbb{G} = \{(y^3 - t, [x^9 - s]), (y - x, [x^3 - t])\}$$

and $\mathbb{D} = \{(\emptyset, \mathcal{S}_1), (\emptyset, \mathcal{S}_2)\}$, where $\mathcal{S}_1 = [x^9 - s, y^3 - t]$ and $\mathcal{S}_2 = [x^3 - t, y - x]$. These two simple sets are returned as output of the algorithm.

5.3.2 Computing radicals of polynomial ideals

Computing a finite basis for the radical of a positive-dimensional polynomial ideal over a finite field is difficult in general. There are three methods due to Matsumoto [106], Kemper [83], and Fortuna and others [62], but no routine available in popular Computer Algebra systems like Maple and Magma for such computation. For our purpose of simple decomposition, the ideal whose radical need be computed is not arbitrary: it is generated by a simple set with one additional polynomial having bigger leading variable. In this part, we investigate the application of the above-mentioned three methods in our particular situation.

Methods of Matsumoto and Kemper

The method developed by Matsumoto [106] is characterized by its use of the Frobenius mapping over $\mathbb{F}_q[\boldsymbol{x}]$, that is, the mapping $f : G \mapsto G^p$. It works for any given ideal $\mathfrak{a} \subset \mathbb{F}_q[\boldsymbol{x}]$ by computing $\mathfrak{a} := f^{-1}(\mathfrak{a})$ recursively until $\mathfrak{a}$ stabilizes. To find the radical of the ideal $\mathfrak{a}_{\tilde{\mathbb{F}}_q} \subset \tilde{\mathbb{F}}_q[\boldsymbol{y}]$ needed for simple decomposition, we can contract $\mathfrak{a}_{\tilde{\mathbb{F}}_q}$ to an ideal $\mathfrak{a}$ in $\mathbb{F}_q[\boldsymbol{x}]$, compute the radical of the contracted ideal $\mathfrak{a}$ using Matsumoto's method, and then extend the radical back to $\tilde{\mathbb{F}}_q[\boldsymbol{y}]$.

Kemper's method [83] was proposed for computing the radical of any zero-dimensional ideal in $\tilde{\mathbb{K}}[\boldsymbol{y}]$. It generalizes the method given in [141] for fields of characteristic 0. The main difficulty for the generalization lies in that over a field of positive characteristic, a squarefree polynomial may no longer remain squarefree when it is extended to a larger field. This difficulty was overcome by means of replacing the squarefree part of a polynomial in the method in [141] by its separable part.

Both of the methods recalled above are based on Gröbner bases computation with introduction of additional variables and polynomials. We can apply these general methods to compute the radical of the particular ideal in our algorithm of simple decomposition without taking the special structure of the ideal into account. In fact, we have not figured out how to make use of the ideal structure to speed up the computation. Our attention will therefore be focused on another method to be discussed, optimized, and improved in the following subsections.

FGT method: brief review

The FGT method proposed by Fortuna and others [62] can be used to compute the radical $\sqrt{\mathfrak{a}}$ of any ideal $\mathfrak{a} \subset \tilde{\mathbb{F}}_q[\boldsymbol{y}]$. It starts with the computation of the radical of $\mathfrak{a} \cap \tilde{\mathbb{F}}_q[y_1]$ (e.g., by using Algorithm SQFREE-FGE in [68]) and then adjoins the next elimination ideal inductively to the radical already computed. Thus the key ingredient of this method is to compute the radical of $\mathfrak{a} \cap \tilde{\mathbb{F}}_q[\boldsymbol{y}_r]$, when $\mathfrak{a} \cap \tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]$ is radical.

For simple decomposition, we need to compute the radical of $\langle \mathcal{S} \cup \{T\} \rangle_{\tilde{\mathbb{F}}_q}$, where $\mathcal{S}$ is a simple set and thus $\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q} = \mathrm{sat}(\mathcal{S})_{\tilde{\mathbb{F}}_q}$ is radical. By Proposition 2.3.4(a) we have $\mathrm{sat}_i(\mathcal{T})_{\tilde{\mathbb{K}}} = \mathrm{sat}(\mathcal{T})_{\tilde{\mathbb{K}}} \cap \tilde{\mathbb{K}}[\boldsymbol{y}_i]$ for each i. Suppose that $\mathrm{lv}(T) = y_s$. Then

$$\langle \mathcal{S} \cup \{T\} \rangle_{\tilde{\mathbb{F}}_q} \cap \tilde{\mathbb{F}}_q[\boldsymbol{y}_{s-1}] = \mathrm{sat}(\mathcal{S} \cup \{T\})_{\tilde{\mathbb{F}}_q} \cap \tilde{\mathbb{F}}_q[\boldsymbol{y}_{s-1}] = \mathrm{sat}(\mathcal{S})_{\tilde{\mathbb{F}}_q} = \langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}.$$

At this point, one sees that our problem is indeed what the FGT method can solve. In what follows, we briefly review the method and optimize it according to our setting.

For any polynomial ring $\mathcal{R}[z]$, we use $'$ to denote the derivation w.r.t. z (i.e., for any $F \in \mathcal{R}[z]$, $F' := \partial F / \partial z$). For any zero-dimensional ideal $\mathfrak{a} = \langle F_1, \ldots, F_r \rangle \subset \mathcal{R}[z]$ and derivation D of $\mathcal{R}[z]$, define $D(\mathfrak{a}) := \langle D(F_1), \ldots, D(F_r) \rangle$. The FGT method is outlined in Algorithm 17 and the reader may refer to [62] for more details.

Algorithm 17: $\sqrt{\mathfrak{a}} = \mathsf{update}(\mathfrak{a})$

Input: $\mathfrak{a}$ — a zero-dimensional ideal in $\mathcal{R}[z]$, with $\mathfrak{a} \cap \mathcal{R} = \mathfrak{b}$ being radical.

Output: $\sqrt{\mathfrak{a}}$ — the radical of $\mathfrak{a}$.

17.1 Compute $\{D_1, \ldots, D_r\}$, a basis of $\mathrm{Der}(\mathcal{R}/\mathfrak{b})$;

17.2 $\mathfrak{l} := \mathfrak{a} : \langle \mathfrak{a}', D_1(\mathfrak{a}), \ldots, D_r(\mathfrak{a}) \rangle$; $\mathfrak{h} := \mathfrak{a} : \mathfrak{l}^{\infty}$;

17.3 Compute $G_1, \ldots, G_s$ and $\mathfrak{j}_1, \ldots, \mathfrak{j}_s$ such that $\mathfrak{h} = \bigcap_{i=1}^{s} \langle G_i, \mathfrak{j}_i \rangle$ and $G_i \equiv H_i^p$ mod $\mathfrak{j}_i$;

17.4 Compute H_i for $i = 1, \ldots, s$;

17.5 **return** $\mathfrak{l} \cap (\bigcap_{i=1}^{s} \mathsf{update}(\langle H_i, \mathfrak{j}_i \rangle))$

In Line 17.1 of Algorithm 17, a basis of $\mathrm{Der}(\mathcal{R}/\mathfrak{b})$ can be computed according to Proposition 3.2.6 and the comments below the proposition. The ideal $\mathfrak{l}$ in Line 17.2 is guaranteed to be radical, while $\mathfrak{h}$ there contains polynomials as pth powers and thus is not radical. The decomposition of $\mathfrak{h}$ in Line 17.3 is related to Gröbner bases under specification and can be achieved as shown in [67]. In each branch of the decomposition, the polynomial G_i is a pth power modulo $\mathfrak{j}_i$, and its pth root is extracted in Line 17.4. The algorithm is completed finally by calling the function update recursively to compute the radical of each $\langle H_i, \mathfrak{j}_i \rangle$.

Algorithm 17 generalizes the algorithm for squarefree decomposition of univariate polynomials over finite fields (see, e.g., [68]) by replacing the ordinary derivative w.r.t. one variable with multiple derivations in $\mathcal{R}/\mathfrak{b}$ and by replacing polynomial quotients with ideal quotients and saturations. The ideals $\mathfrak{l}$ and $\mathfrak{h}$ correspond to the two parts of the squarefree decomposition of a univariate polynomial: one does not contain pth power factors and the other does and need be handled by using additional techniques.

FGT method: optimization

We want to compute efficiently the radical of $\langle \mathcal{S} \cup \{T\} \rangle_{\tilde{\mathbb{F}}_q}$, where $T \in \tilde{\mathbb{F}}_q[\boldsymbol{y}_r]$, and $\mathcal{S} \subset \tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]$ such that $\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$ is radical. For this end, we proceed to optimize the algorithm update for $\langle \mathcal{S} \cup \{T\} \rangle_{\tilde{\mathbb{F}}_q}$ as input by making use of the structure of the special type of input ideals.

(1) For any ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ as input to the algorithm update, a maximal independent subset $\{x_{i_1}, \ldots, x_{i_s}\}$ of variables such that $\mathfrak{a} \cap \mathbb{K}[x_{i_1}, \ldots, x_{i_s}] = \{0\}$ has to be found so as to turn $\mathfrak{a}$ to a zero-dimensional ideal. However, for any regular set whose

parameters can be easily identified, no extra computation is needed.

(2) Recall that p is the characteristic of $\mathbb{F}_q$. The following prejudgement criteria can be used to avoid some useless computations after $\mathfrak{l}$ is obtained.

(a) If $\mathfrak{a} = \mathfrak{l}$, then $\mathfrak{a} = \sqrt{\mathfrak{a}}$. This is because $\mathfrak{l}$ is radical [62, Porposition 3.7].

(b) If $p > \deg(T, y_s)$, then for every associate prime $\mathfrak{p}$ of $\mathrm{sat}(\mathcal{S})_{\tilde{\mathbb{F}}_q}$, $\deg(\overline{T}^{\mathfrak{p}}, y_r) < \deg(T, y_r) < p$ and thus $(\overline{T}^{\mathfrak{p}})' \neq 0$, which implies that $\overline{T}^{\mathfrak{p}}$ is not a pth power. In this case, the ideal $\mathfrak{l}$ is equal to $\sqrt{\mathfrak{a}}$ because the ideal $\mathfrak{h}$ containing pth powers is trivial.

The above criteria identify the cases when some "troublesome" part in the squarefree decomposition of T does not occur and thus the involved computation is unnecessary. Criterion (b) is particularly effective when p is large.

(3) When the generators of $\mathfrak{h}$ form a triangular set, the splitting of $\mathfrak{h}$ in Line 17.3 is no longer needed. This can be shown by using the proof of Corollary 3.8 in [62].

Now we explain how to extract pth roots. Let $P \in \tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}][y_r]$ be a polynomial which is a pth power modulo a radical ideal $\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q} \subset \tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]$. As $P' = 0$, one can write $P = \sum_j P_j y_r^{pj}$, where $P_j \in \tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$. Therefore, the extraction of the pth root of P reduces to that of each P_j in $\tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$. In other words, we only need to deal with the following problem: given a pth power $G \in \tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$, extract its pth root $H \in \tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$.

Proposition 5.3.7. *Let $G \in \tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$ be a pth power, where $\mathcal{S}$ is a simple set. Then the pth root of G in $\tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$ is unique.*

Proof. Suppose that H_1 and H_2 are two pth roots of G in $\tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$. Then $H_1^p - H_2^p = (H_1 - H_2)^p = 0$. As in $\tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$ there is no nilpotent element, we have $H_1 - H_2 = 0$ and thus the conclusion follows. □

Take a basis $\{b_1, \ldots, b_k\}$ of $\tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$ as an $\tilde{\mathbb{F}}_q$-module and write $b_i^p = \sum_{j=1}^k a_{ij} b_j$ modulo $\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$ for $i = 1, \ldots, k$. Suppose that $G = \sum_{j=1}^k g_j b_j$ for $g_j \in \tilde{\mathbb{F}}_q$ and its pth root $H = \sum_{i=1}^k c_i b_i$ with $c_i \in \tilde{\mathbb{F}}_q$ is to be determined. Then in $\tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{F}}_q}$,

$$H^p = \sum_{i=1}^k c_i^p b_i^p = \sum_{i=1}^k c_i^p \left(\sum_{j=1}^k a_{ij} b_j \right) = \sum_{j=1}^k \left(\sum_{i=1}^k a_{ij} c_i^p \right) b_j = \sum_{j=1}^k g_j b_j = G,$$

from which the following linear equations over $\tilde{\mathbb{F}}_q$ can be derived:

$$\sum_{i=1}^{k} a_{ij} c_i^p = g_j, \qquad j = 1, \ldots, k, \tag{5.7}$$

where c_i^p are unknowns.

If the coefficient matrix (a_{ij}) is invertible, then the system of linear equations (5.7) can be directly solved and c_i may be recovered via pth root extraction in $\tilde{\mathbb{F}}_q = \mathbb{F}_q(\boldsymbol{u})$. Otherwise, we can apply the techniques in Condition P of [141, Section 40] to solve (5.7). Theoretically, it has been proved that Condition P is a necessary and sufficient condition on the coefficient field for the radical of an ideal to be computable [62]. Since Condition P holds for the coefficient field $\mathbb{F}_q(\boldsymbol{u})$, we can solve the linear system (5.7) effectively. The solution obtained furnishes the unique pth root of G.

FGT method: specialized algorithm

The previous discussions are combined to devise the algorithm SimpleRadical. It computes $\sqrt{\langle \mathcal{S} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q}}$ for any simple set $\mathcal{S} \subset \tilde{\mathbb{F}}_q[\boldsymbol{y}_{r-1}]$ and polynomial $P \in \tilde{\mathbb{F}}_q[\boldsymbol{y}_{s-1}][y_s]$. We first use a small example to illustrate this algorithm and the FGT method.

Let $\mathfrak{a} = \langle x_1^3 - u, x_2^3 - u \rangle \subseteq \mathbb{F}_3(u)[x_1, x_2]$. The FGT method starts by computing the radical of $\langle x_1^3 - u \rangle \subseteq \mathbb{F}_3(u)[x_1]$ using Algorithm SQFREE-FGE in [68], resulting in $\sqrt{\langle x_1^3 - u \rangle} = \langle x_1^3 - u \rangle$. Then according to Proposition 3.2.6, a basis of $\mathrm{Der}(\mathbb{F}_3(u)[x_1]/\langle x_1^3 - u \rangle)$ is computed as $\partial/\partial x_1, \partial/\partial x_2$. Thus $\mathfrak{l} = \langle 1 \rangle$ and $\mathfrak{h} = \mathfrak{a}$. Since the generators of $\mathfrak{h}$ already form a regular set, no splitting is needed for $\mathfrak{h}$. It is verified in this step that $x_2^3 - u$ is a pth power modulo $\langle x_1^3 - u \rangle$.

The pth root of $x_2^3 - u$ is extracted as follows. In order to compute the pth root of the coefficient $2\,u$, the following system of linear equations corresponding to (5.7) is constructed:

$$\begin{pmatrix} 1 & u & u^2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} c_0^p \\ c_1^p \\ c_2^p \end{pmatrix} = \begin{pmatrix} 2\,u \\ 0 \\ 0 \end{pmatrix}.$$

The coefficient matrix is clearly not invertible and Condition P should be applied. According to the procedure described in [141, Section 40] with independent set $[1, u, u^2]$,

the above system can be expanded further as

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix},$$

whose solution is $c_1 = 2$ and $c_0 = c_2 = 0$. This means that the pth root of $2\,u$ is $2\,x_1$. Therefore, we obtain the pth root $x_2 - x_1$ of $x_2^3 - u$.

Then $\sqrt{\langle x_1^3 - u, x_2 - x_1 \rangle}$ is computed recursively. Now the ideal is already radical because $\deg(x_2 - x_1, x_2) = 1 < p$, so the computation of $\sqrt{\mathfrak{a}}$ ends.

For this example, the ideal $\mathfrak{l}$ is trivial and no further splitting of $\mathfrak{h}$ is needed, but it is not so in general.

Algorithm 18: $\mathfrak{l} := \mathsf{SimpleRadical}(P, \mathcal{B})$

Input: $\mathcal{B}$ — a zero-dimensional simple set in $\tilde{\mathbb{F}}_q[\boldsymbol{y}_{s-1}]$; P — a polynomial in $\tilde{\mathbb{F}}_q[\boldsymbol{y}_s]$.

Output: $\sqrt{\langle \mathcal{B} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q}}$ — the radical of $\langle \mathcal{B} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q}$.

18.1 Compute $\{D_1, \ldots, D_u\}$, a basis of $\mathrm{Der}(\tilde{\mathbb{F}}_q[\boldsymbol{y}_{s-1}]/\langle \mathcal{B} \rangle_{\tilde{\mathbb{F}}_q})$;

18.2 $\mathfrak{l} := \langle \mathcal{B} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q} : \langle \langle \mathcal{B} \cup \{P\} \rangle'_{\tilde{\mathbb{F}}_q}, D_1(\langle \mathcal{B} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q}), \ldots, D_u(\langle \mathcal{B} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q}) \rangle$;

18.3 **if** $p > \deg(P, y_s)$ *or* $\mathfrak{l} = \langle \mathcal{B} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q}$ **then**

18.4 | **return** $\mathfrak{l}$;

18.5 **else**

18.6 | $\mathfrak{h} := \langle \mathcal{B} \cup \{P\} \rangle_{\tilde{\mathbb{F}}_q} : \mathfrak{l}^\infty$;

18.7 | Compute $G_1, \ldots, G_v$ and $\mathfrak{j}_1, \ldots, \mathfrak{j}_v$ such that $\mathfrak{h} = \bigcap_{i=1}^{v} \langle G_i, \mathfrak{j}_i \rangle$ and each G_i is a pth power modulo $\mathfrak{j}_i$;

18.8 | **for** $i = 1, \ldots, v$ **do**

18.9 | | Suppose that $\mathfrak{j}_i = \langle \mathcal{B}_i \rangle_{\tilde{\mathbb{F}}_q}$, where $\mathcal{B}_i$ is a simple set, and $G_i = \sum_j G_{ij} y_s^{pj}$ with $G_{ij} \in \tilde{\mathbb{F}}_q[\boldsymbol{y}_{s-1}]/\mathfrak{j}_i$;

18.10 | | Compute a basis $\boldsymbol{b}$ of $\tilde{\mathbb{F}}_q[\boldsymbol{y}_{s-1}]/\mathfrak{j}_i$;

18.11 | | Express $\boldsymbol{b}^p$ as $\tilde{\mathbb{F}}_q$-linear combinations of $\boldsymbol{b}$;

18.12 | | **for** *each* G_{ij} **do**

18.13 | | | Construct the linear system $M\boldsymbol{x} = \boldsymbol{g}$ as (5.7), where $\boldsymbol{g}$ is the column coordinate vector of G_{ij} w.r.t. $\boldsymbol{b}$;

18.14 | | | **if** M *is invertible* **then**

18.15 | | | | Solve $M\boldsymbol{x} = \boldsymbol{g}$ for $\boldsymbol{x}$ to obtain $\overline{\boldsymbol{x}}$; $\overline{\boldsymbol{x}} := \sqrt[p]{\overline{\boldsymbol{x}}}$;

18.16 | | | **else**

18.17 | | | | Use Condition P to obtain $\overline{\boldsymbol{x}}$ such that $M\overline{\boldsymbol{x}}^p = \boldsymbol{g}$;

18.18 | | | **end**

18.19 | | | $H_j := \overline{\boldsymbol{x}} \cdot \boldsymbol{b}$;

18.20 | | **end**

18.21 | | $\mathfrak{h}_i = \mathsf{SimpleRadical}(\sum_j H_j y_s^j, \mathcal{B})$;

18.22 | **end**

18.23 | **return** $\mathfrak{l} \cap (\bigcap_{i=1}^{v} \mathfrak{h}_i)$

18.24 **end**

5.4 Implementation and experimental results

We have implemented the algorithms for decomposing zero- and positive-dimensional polynomial sets into simple sets over finite fields mainly in Maple using the RegularChains library [93], which is useful for computations related to regular sets. As currently this library can only handle polynomial sets over prime finite fields, i.e. $\mathbb{F}_p$, our implementation also has this restriction. In this section we present the experimental results obtained for all our implementations and provide concise comparison and analysis.

5.4.1 Zero-dimensional polynomial sets

Various regular sets as shown in Table 5.2 are tested, covering fields of small, medium and big characteristics ($\mathbb{F}_5$, $\mathbb{F}_{541}$ and $\mathbb{F}_{7919}$), to verify the effectiveness of our algorithms. Preliminary observations and analyses are given below, which may shed light on the differences of the three algorithms and suggest which one to choose in different situations.

All the experiments were made in Maple 11 running on AMD Athlom(tm) II X2 CPU 1.60GHz with 2.00G RAM under Windows XP OS. Table 5.1 records the timing (in seconds) of each algorithm, followed by the branch number of the output in brackets. The last column gives the prejudging time in SimDecZeroPJ.

Table 5.1: Timings of decomposing regular sets into simple sets (zero-dimensional)

No.	SSimDecZero	SimDecZero	SimDecZeroPJ	Prejudge
P1	0.766 (21)	189.984 (16)	191.610 (16)	2.985
P2	331.750 (28)	392.437 (20)	163.891 (20)	16.094
P3	72.578 (72)	97.922 (36)	43.610 (36)	0.202
P4	18.922 (36)	16.907 (18)	11.687 (18)	7.890
P5	85.625 (24)	133.188 (12)	133.906 (12)	1.421
P6	96.187 (36)	187.047 (36)	76.641 (36)	8.641
P7	59.547 (108)	542.672 (180)	166.250 (180)	0.000
P8	924.687 (8)	653.297 (6)	382.391 (6)	0.000
P9	286.860 (4)	387.453 (2)	173.984 (2)	0.000
P10	80.812 (2)	84.016 (1)	0.890 (1)	0.797
P11	228.672 (10)	228.750 (12)	0.391 (12)	0.000
P12	585.844 (120)	590.406 (120)	497.250 (120)	0.000
P13	267.656 (14)	229.844 (16)	0.484 (16)	0.000
P14	851.406 (48)	1658.016 (11)	404.765 (11)	0.000

From the experiments, one may observe the following.

- The strong simple decomposition algorithm is more efficient than the weak one in most cases and especially for P1, P7, and P14. This may be due to the influence of the complexity of polynomials in a regular set $\mathcal{T}$ on the performance of $\mathsf{pgcd}(*, \mathcal{T})$ computation. Compared with SimDecZero, the algorithm SSimDecZero reduces the complexity of polynomials in the simple sets, on which later pseudo-gcd computation is performed.

- As shown by P4, the prejudging process may be quite time-consuming compared with the squarefree part computation.

- If the prejudging process can detect some branches for which $F/\gcd(F, F')$, rather than the complete squarefree decomposition, is sufficient for computing the squarefree part, then the algorithm SimDecZeroPJ may save a lot of time in obtaining the squarefree part of polynomials. See P3, P4, P9, P10, P11, and P13 for instance. In this case, the amount of saved time depends on the time spent in the **while** loop of Algorithm 14.

5.4.2 Positive-dimensional polynomial sets

Due to the lack of function in Maple for syzygy and ideal-saturation computation over finite fields, our implementation has to call external functions from Singular (Version 3-1-4) [41] for some of the computations. Communication between the two systems for our purpose is automated. For the sake of comparison, we have also implemented in Maple the methods of Matsumoto and Kemper for computing radicals of positive-dimensional ideals over finite fields.

We have carried out experiments with test examples for our algorithm of simple decomposition using the three methods discussed in Section 5.3.2 for the involved computation of radicals. Here we present the experimental results for 18 positive-dimensional regular sets (Q1–Q18 in Table 5.5), with finite fields of characteristics $3, 17$, and 53 used. All the experiments were made on an Intel(R) Pentium(R) P8700 CPU 2.53 GHz with 1.89 G RAM under Windows XP Professional SP3.

Table 5.3 shows the timings (in seconds) of simple decomposition based on the three methods of Matsumoto, Kemper and FGT for radical computation in Maple

and the method of Kemper for radical computation in Singular. In the table, > 3600 indicates that the computation did not terminate within one hour and "overflow" means that the computation stopped abnormally because of overflow. The timings show that the algorithm implemented in Maple using the FGT method is faster in general than that using the method of Matsumoto or Kemper, in particular when the prejudgement criteria hold (e.g., for Q11). As the condition $p > \deg(T, y_s)$ is expected to hold with high probability for fields of big characteristics, it may be concluded that the FGT-method-based algorithm performs better than the algorithm based on the two other methods for big characteristics.

Without optimization, the methods of Matsumoto and Kemper are comparable in terms of efficiency for those special ideals arising from simple decomposition. Mastumoto's method is more sensitive to the number of parameters and its efficiency decreases dramatically as the number of parameters increases (see, e.g., Q5 and Q7), while the efficiency of Kemper's method depends more on the characteristics of the ground field (see, e.g., Q5 and Q13).

However, our implementation of the algorithm for simple decomposition using Kemper's method (or even using the FGT method) in Maple performs much worse than that using Kemper's method implemented (by Gerhard Pfister) in Singular, as shown in Table 5.3, Columns 5 and 6. This is mainly due to the computation of elimination ideals using the EliminitionIdeal function from the PolynomialIdeals package, which is very costly in Maple. Table 5.4 shows the total times for computing the elimination ideals in Maple (Column 3) and for computing the radicals in Maple (Column 4) and in Singular (Column 5) using Kemper's method for the same set of 18 examples. One sees clearly that computation of elimination ideals takes the large majority of the total computing times. Therefore, our implementation of the algorithm in Maple may become much more efficient if a more efficient function is implemented to replace EliminationIdeal for computing elimination ideals.

The compactness of representations of simple sets may also be confirmed experimentally: for Q6, the simple decomposition computed by using our algorithm has 5 components, whereas the prime or irreducible decomposition (computed, e.g., by using the function minAssChar in Singular) has 27 components.

The algorithm proposed for positive-dimensional polynomial sets works also for

zero-dimensional ones (when the coefficient matrix of (5.7) is always invertible, as shown by Proposition 5.2.6 and its following remark). We have applied the algorithm to the zero-dimensional benchmarks given in Table 5.2. Some of the empirical results are reported in [99], showing the performance of the algorithm in the zero-dimensional case. However, in regard to efficiency, those specially designed algorithms for zero-dimensional polynomial sets outperform the more general one for positive-dimensional polynomial sets.

In addition to the output radical ideal, Algorithm 18 computes some other radical ideals (e.g., the ideals $\mathfrak{l}$ in all recursive calls of the algorithm), whose intersection is the output ideal. This observation allows one to introduce a variant of the algorithm. Instead of returning the radical ideal, the algorithm may output the sequence of ideals $\mathfrak{l}$. Handling such ideals by using the decomposition technique explained in Section 5.3, one may obtain another simple decomposition. It may contain more branches than the one returned by Algorithm 18 and thus is less compact in terms of representation, while the average size of the branches can become smaller. In fact, by means of factorizing intermediate polynomials, one can split simple sets (or even decompose them into irreducible triangular sets, leading to prime decompositions). The splitting reduces the size of polynomials and thus may help speed up the process of decomposition, but the representation of the resulting decompositions may become less compact. As we have already mentioned, our interest is in computing simple decompositions with compact representations, so polynomial factorization is not used and splitting takes place only when it is really necessary.

Table 5.2: Regular set benchmarks (zero-dimensional)

No.	char	Regular sets ($x_1 < x_2 < x_3 < x_4 < x_5$)
P1	5	$[x_1(x_1-2)^6, x_2^2(x_2-1), (x_3-1)^2(x_3-2)^3, (x_4-1)(x_4-2)^5, ((x_1-1)\,x_5^6 + x_1x_2x_5^2 + (x_1-2)(x_2-1)(x_3-1)\,x_4x_5^3+1)(x_5+x_4+x_3)]$
P2	5	$[x_1^5(x_1+1)^{17}(x_1+2)^9, (x_1^2+x_1+1)\,x_2^5+(2\,x_1^3+3)\,x_2+3, ((x_1^4+x_2)\,x_3+x_2)^5((x_2+1)\,x_3^5+(x_1+x_3)\,x_3^2+1)^7, (x_4^{11}+2\,x_4+4)^{11}(x_4+3)^{23}, (x_1+x_2+x_3)\,x_5^3+(x_1+x_2)\,x_5+(x_3+x_4)]$
P3	5	$[(x_1^4+4\,x_1^2+3\,x_1+1)(x_1^5+4\,x_1+2)(x_1+3)^{15}, (x_2^4+x_2^2+2)^2(x_2^3+x_2+4)^{10}, (x_3^2+x_3+2)^3(x_3^3+x_3^2+x_3)^{15}, (x_4-2)^5(x_4-1), ((x_1-x_2)\,x_5+1)^2(x_5+x_4+x_3)]$
P4	5	$[(x_1^4+4\,x_1^2+3\,x_1+1)^3(x_1^5+4\,x_1+2)(x_1+3), (x_2^4+x_2^2+2)^2(x_2^3+x_2+4)^7, (x_3^2+x_3+2)^3(x_3^3+x_3^2+x_3)^{11}, (x_4-2)^7, ((x_1-1)\,x_5+1)^4(x_5+x_4+x_3)]$
P5	5	$[(x_1^4+4\,x_1^2+3\,x_1+1)(x_1+3)^{15}, (x_2^4+x_2^2+2)^2(x_2^3+x_2+4)^{10}, (x_3^2+x_3+2)^3(x_3^3+x_3^2+x_3)^{15}, (x_4-2)^5, ((x_1-1)\,x_5^6+1)^2(x_5+x_4+x_3)]$
P6	5	$[x_1(x_1-2)^5, (x_2^2+2)^2(x_2^3+4\,x_2^3+3\,x_2+1)^3, (x_3^4+4\,x_3^2+3\,x_3+1)(x_3^3+x_3^2+1)^2(x_3-2)^3, (x_4-2)^5(x_4^5+2\,x_4^3+3\,x_4^2+x_4+4)(x_4^4+3\,x_4^2+x_4+1)^3, ((x_1-1)\,x_5^6+x_1x_2x_5^2+(x_1-2)(x_2-1)(x_3-1)\,x_4x_5^3+1)]$
P7	541	$[x_1(x_1-2)^5, x_2^2(x_2^3+4\,x_2^3+3\,x_2+1)(x_2^4+3\,x_2^2+3)^3, (x_3^4+4\,x_3^3+3\,x_3+1)(x_3^3+x_3^2+1)^2(x_3-2)^3, (x_4-2)^5(x_4^5+2\,x_4^3+3\,x_4^2+x_4+4)(x_4^4+3\,x_4^2+x_4+1)^3, (x_1-1)\,x_5^{15}+(x_4+x_1)\,x_5^{10}]$
P8	541	$[(234\,x_1^2+257)(153\,x_1+412)^3, ((x_1+23)\,x_2^3+(23\,x_1^3+264)\,x_2+521)(267\,x_2^3+123)^7, 255\,x_3^4+345\,x_3+112, (234\,x_1^3+341\,x_2^2+194\,x_3)\,x_4^2+(x_1+x_2+x_3), (283\,x_1+203\,x_2+461\,x_3+123\,x_4)\,x_5^6+(234\,x_1^3+x_1)\,x_5^2+237]$
P9	541	$[(12\,x_1^2+62)^{155}(153\,x_1+412)^3, ((x_1+23)\,x_2^3+(23\,x_1^3+264)\,x_2+521)^{13}((23\,x_1^3+236)\,x_2^3+123)^{27}, 13\,x_3+531, 43\,x_4^2+342\,x_4+249, 345\,x_5+82]$
P10	541	$[(323\,x_1+52)^{432}(x_1^2+236)^{117}, x_2, x_3, x_4, x_5]$
P11	7919	$[(x_1^4+4\,x_1^2+3\,x_1+1)^{31}(x_1^5+4\,x_1+2)^{11}(x_1+3)^{329}(x_1-4)^{537}, x_2-1, x_3^2+x_3+2, x_4-2, (x_1-1)\,x_5+1]$
P12	7919	$[(1234\,x_1^{13}+1435\,x_1^7+4576)(323\,x_1^5+134\,x_1^4+2356)^9, (2346\,x_2^3+345\,x_2^3+865)(234\,x_2^2+2456)^3, 645\,x_3^3+6346\,x_3, (1234\,x_4-345)^{11}(234\,x_4^2+345\,x_4+2346)^7, (376\,x_1^3-2134\,x_2)\,x_5^{27}+4565\,x_5^{12}+255$
P13	7919	$[(1244\,x_1^4+6454\,x_1^2+3465\,x_1+5345)^{31}(155\,x_1^5+4545\,x_1+235)^{11}(215\,x_1+3125)^{329}(2356\,x_1-4123)^{537}, 346\,x_2-1214, 1234\,x_3^2+214\,x_3+2234, 2423\,x_4-234, (2443\,x_1^5-456\,x_4)\,x_5+2134]$
P14	7919	$[(2445\,x_1^3+3456)^5(235\,x_1+767)^7, ((156\,x_1^2+124)\,x_2^3+266\,x_2+1676)^3(235\,x_2^4+3671\,x_2^3+(234\,x_1+31))^5, (234\,x_1^2+23\,x_2)\,x_3^3+235, (13\,x_4+235)^5(235\,x_4+3467\,x_3+272\,x_2+3678\,x_1)^7, (435\,x_5^7+2347\,x_5^3+1236)^7(2734\,x_5+234)^3]$

Table 5.3: Timings for decomposing regular sets into simple sets (positive-dimensional)

Ex	Char	FGT/Maple	Matsumoto/Maple	Kemper/Maple	Kemper/Singular
Q1	3	0.484	0.547	4.204	0.415
Q2	3	69.937	0.515	>3600	0.467
Q3	3	0.109	9.657	63.797	0.169
Q4	3	0.516	104.047	205.421	0.223
Q5	3	1.578	3.578	14.86	0.779
Q6	3	18.843	>3600	174.812	1.931
Q7	3	81.609	>3600	720.015	3.064
Q8	3	42.813	>3600	>3600	4.512
Q9	17	0.078	1.656	78.453	overflow
Q10	17	0.172	>3600	0.422	0.223
Q11	17	0.203	>3600	206.407	0.114
Q12	17	0.812	2.313	>3600	0.645
Q13	17	592.781	41.297	>3600	21.138
Q14	53	0.047	>3600	>3600	overflow
Q15	53	0.219	827.235	0.203	0.208
Q16	53	62.297	0.656	>3600	1.614
Q17	53	54.968	11.859	>3600	46.635
Q18	53	7.11	460.921	>3600	6.953

Table 5.4: Timings for radical computation in Maple and Singular

Ex	Char	EliminitionIdeal/Maple	Radical/Maple	Radical/Singular
Q1	3	3.734	3.859	0.04
Q2	3	>3600	>3600	0.03
Q3	3	11.5	63.718	0.06
Q4	3	205.171	205.25	0.02
Q5	3	14.11	14.218	0.06
Q6	3	172.75	173.078	0.04
Q7	3	609.563	609.828	0.08
Q8	3	>3600	>3600	0.09
Q9	17	16.547	78.375	overflow
Q10	17	0.11	0.203	0.02
Q11	17	206.125	206.203	0.02
Q12	17	>3600	>3600	0.02
Q13	17	>3600	>3600	2.06
Q14	53	>3600	>3600	overflow
Q15	53	0.031	0.063	0.02
Q16	53	>3600	>3600	0.02
Q17	53	>3600	>3600	0.01
Q18	53	>3600	>3600	0.11

Table 5.5: Regular set benchmarks (positive-dimensional)

Ex	Regular set ($u_1 < u_2 < u_3 < x_1 < x_2 < x_3$)
Q1	$[(x_1^3 - u_1)(x_1^9 - u_2), x_2^3 - u_1]$
Q2	$[(x_1^{3^5} - u_1)(x_1^9 - u_2), x_2^3 - u_1]$
Q3	$[x_1^{3^{10}} - u_1^{3^{10}}]$
Q4	$[u_1x_1^5 - u_2, u_2x_2^5 - u_1, u_3x_3^5 - u_1 + u_2]$
Q5	$[(x_1^3 - u_1)(x_1^3 - u_2), (x_2^3 - u_1)(x_2^3 - u_2), (x_3^3 - u_1)(x_3^3 - u_2)]$
Q6	$[(x_1^3-u_1)(x_1^3-u_2)(x_1^3-1), (x_2^3-u_1)(x_2^3-u_2)(x_2^3-1), (x_3^3-u_1)(x_3^3-u_2)(x_3^3-1)]$
Q7	$[(x_1^3-u_1)(x_1^3-u_2)(x_1^3-u_3), (x_2^3-u_1)(x_2^3-u_2)(x_2^3-1), (x_3^3-u_1)(x_3^3-u_2)(x_3^3-1)]$
Q8	$[(x_1^3-u_1)(x_1^3-u_2)(x_1^3-u_3), (x_2^3-u_1)(x_2^3-u_2)(x_3^3-u_3), (x_3^3-u_1)(x_3^3-u_2)(x_3^3-u_3)]$
Q9	$[x_1^{17^4} - u_1^{17^4}]$
Q10	$[x_1^5 - u_1x_1^2 + u_1, x_2^5 - u_2x_1 + u_1]$
Q11	$[u_1x_1^5 - u_2, u_2x_2^5 - u_1, u_3x_3^5 - u_1 + u_2]$
Q12	$[(x_1^{17} - u_1)(x_1^{17} - u_2), (x_2^{17} - u_1)(x_2^{17} - u_2)]$
Q13	$[(x_1^{17} - u_1)(x_1^{17} - u_2), (x_2^{17} - u_1)(x_2^{17} - u_2), (x_3^{17} - u_1)(x_3^{17} - u_2)]$
Q14	$[x_1^{53^5} - u_1^{53^5}]$
Q15	$[x_1^3 - u_1x_1^2 + u_1, x_2^3 - u_1x_1]$
Q16	$[(x_1^{53} - u_1)(x_1^{53} - u_2), x_2^{53} - u_1]$
Q17	$[(x_1^{53} - u_1)(x_1^{53} - 1), (x_2^{53} - u_1)(x_2^{53} - 1)]$
Q18	$[(x_1^{53} - u_1)(x_1^{53} - u_2), (x_2^{53} - u_1)(x_2^{53} - u_2)]$

Chapter 6

Squarefree decomposition and factorization over unmixed products of field extensions

The problems of squarefree decomposition and factorization of polynomials have been well studied in the area of computer algebra. Algorithms developed for their solutions, in particular for polynomials over finite fields, the field of rational numbers, and their transcendental and algebraic extensions [68, 79, 123, 160, 75, 146], have been implemented to back efficient functioning of many core routines in generations of computer algebra systems. In this chapter we address the problems and provide algorithmic solutions to them for polynomials over unmixed products of field extensions, which are not necessarily unique factorization domains (UFDs), but have an algebraic structure close to that of UFDs.

In this chapter we identify unmixed products of field extensions which correspond to and can be represented by simple sets, and introduce the concepts of squarefree decomposition, irreducibility, and factorization of polynomials over unmixed products, Then we propose algorithms for squarefree decomposition and factorization of polynomials over unmixed products, with illustrative examples and preliminary experiments with our initial implementation.

We define unmixed products of field extensions, connect them to simple sets, and list a few basic operations over unmixed products in Section 6.1. Then we present three algorithms for squarefree decomposition and factorization of polynomials over

unmixed products using multiple derivations and transformations of ideals to shape position in Sections 6.2 and 6.3. Some examples and preliminary experimental results are provided in Section 6.4 to illustrate our algorithms and to show their performance.

The results in this chapter are based on the joint work with Dongming Wang with a paper in preparation.

6.1 Unmixed products of field extensions

6.1.1 Representations

In Section 5.1, we have identified the relationship between simple sets and products of field extensions preliminarily. Now we elaborate which kind of products of field extensions simple sets correspond to.

Let $i \leq j$ be positive integers. For any field $\mathbb{K}$, we denote by $\pi_{j,i}$ the projection from $\mathbb{K}^j$ to $\mathbb{K}^i$, which maps $(a_1, \ldots, a_j)$ to $(a_1, \ldots, a_i)$. A zero-dimensional variety $V \subset \mathbb{K}^j$ is said to be *equiprojectable* if $\pi_{j,i}^{-1}(M_1)$ and $\pi_{j,i}^{-1}(M_2)$ have the same cardinality for any $M_1, M_2 \in \pi_{j,i}(V)$ and $1 \leq i \leq j$.

By Theorem 4.5 in [7], any zero-dimensional variety over a perfect field $\mathbb{K}$ is equiprojectable if and only if its defining ideal is generated by a simple set over $\mathbb{K}$.

Definition 6.1.1. A product of field extensions $\mathbb{K}_1, \ldots, \mathbb{K}_t$ over $\mathbb{K}$ is said to be *unmixed* if

(a) $\mathbb{K}_i$ can be written in the form

$$\mathbb{K}(u_1, \ldots, u_s)(\beta_{i,1}, \ldots, \beta_{i,r})$$

with same $u_1, \ldots, u_s$ and $r + s = n$ for all $i = 1, \ldots, t$, where $u_1, \ldots, u_s$ are transcendental over $\mathbb{K}$ and $\beta_{i,1}, \ldots, \beta_{i,r}$ are algebraic over $\tilde{\mathbb{K}} = \mathbb{K}(u_1, \ldots, u_s)$;

(b) the maximal ideals $\mathfrak{m}_i$ and $\mathfrak{m}_j$ such that $\mathbb{K}_i = \tilde{\mathbb{K}}[\boldsymbol{y}]/\mathfrak{m}_i$ and $\mathbb{K}_j = \tilde{\mathbb{K}}[\boldsymbol{y}]/\mathfrak{m}_j$ are coprime for all $i \neq j$;

(c) the union V of the varieties of $\mathfrak{m}_i$ $(i = 1, \ldots, t)$ in $\bar{\mathbb{K}}^r$ is equiprojectable, where $\bar{\mathbb{K}}$ denotes the algebraic closure of $\tilde{\mathbb{K}}$.

For any simple set $\mathcal{S} \subset \mathbb{K}[\boldsymbol{x}]$, $\tilde{\mathbb{K}}[\boldsymbol{y}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{K}}} = \tilde{\mathbb{K}}[\boldsymbol{y}]/\operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{K}}}$ is isomorphic to a product of field extensions such that Definition 6.1.1 (a) and (b) hold, with $u_1, \ldots, u_s$

being the parameters of $\mathcal{S}$. By the proof technique used for Theorem 4.5 in [7], we can show that condition (c) holds as well.

On the other hand, for any unmixed product of field extensions $\mathbb{K}_1, \ldots, \mathbb{K}_t$ as in Definition 6.1.1, there exists a simple set $\mathcal{S} \subset \mathbb{K}[\boldsymbol{x}]$ such that $\langle \mathcal{S} \rangle_{\tilde{\mathbb{K}}}$ is the ideal defined by V. The proof of this statement is similar to that of Theorem 4.5 in [7]. By Definition 6.1.1 (b) and (c) and the Chinese Remainder Theorem, we have $\tilde{\mathbb{K}}[\boldsymbol{y}]/\langle \mathcal{S} \rangle_{\tilde{\mathbb{K}}} \cong \prod_{i=1}^{t} \tilde{\mathbb{K}}[\boldsymbol{y}]/\mathfrak{m}_i$. Therefore, every simple set determines an unmixed product of field extensions, and vice verse.

Simple sets are convenient representations for unmixed products of field extensions. When speaking about a simple set $\mathcal{S}$ representing an unmixed product A, we mean that the relationship $\tilde{\mathbb{K}}[\boldsymbol{y}]/\operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{K}}} \cong A$ holds. At this point one can see that performing operations in A is equivalent to performing computations in $\tilde{\mathbb{K}}[\boldsymbol{y}]$ modulo $\operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{K}}}$.

As explained in Section 5.1, one advantage of using simple sets is that related field extensions can be represented together. This makes the representations compact and thus may lead to potential improvement on the performance of manipulation with the field extensions.

6.1.2 Basic operations over unmixed products

Let A be an unmixed product of field extensions $\mathbb{K}_1, \ldots, \mathbb{K}_t$ of $\mathbb{K}$, represented by a simple set $\mathcal{S} \subset \mathbb{K}[\boldsymbol{x}]$, and let $\tilde{\mathbb{K}}$ denote $\mathbb{K}(\boldsymbol{u})$. These notations will be fixed in what follows. Next we review basic operations over unmixed products of field extensions. The readers may refer to [78, 38] for more details such as their complexity analyses.

Representation in A. Any element $f \in A$ can be written as $f = (f_1, \ldots, f_t)$ with $f_i \in \mathbb{K}_i$ for $i = 1, \ldots, t$. Note that each f_i can be represented as a polynomial in $\beta_{i,1}, \ldots, \beta_{i,r}$ with coefficients in $\tilde{\mathbb{K}}$. Let F_i be the polynomial in $\tilde{\mathbb{K}}[\boldsymbol{y}]$ obtained from f_i by replacing $\beta_{i,j}$ with y_j and let $\mathcal{T}_i$ be the irreducible triangular set such that $\mathbb{K}_i \cong \tilde{\mathbb{K}}[\boldsymbol{y}]/\operatorname{sat}(\mathcal{T}_i)_{\tilde{\mathbb{K}}}$. By the Chinese Remainder Theorem (Theorem 3.1.13) there exists a unique polynomial $F \in \tilde{\mathbb{K}}[\boldsymbol{y}]$ modulo $\operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{K}}}$ such that $F \equiv F_i \mod \operatorname{sat}(\mathcal{T}_i)_{\tilde{\mathbb{K}}}$ for $i = 1, \ldots, t$.

We use the normal form of F w.r.t. the Gröbner basis of $\langle \mathcal{S} \rangle_{\tilde{\mathbb{K}}} = \operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{K}}}$ to represent $f \in A$. Therefore, each $f \in A$ corresponds to a unique representation in

$\tilde{\mathbb{K}}[\boldsymbol{y}]/\operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{K}}}$, and vice verse.

Arithmetic in A. For any $f, g \in A$ represented by $F, G \in \tilde{\mathbb{K}}[\boldsymbol{y}]$ respectively, we compute $F \pm G$ and FG in $\tilde{\mathbb{K}}[\boldsymbol{y}]/\operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{K}}}$ as the representations of $f \pm g$ and fg (which are obviously in A) respectively. The representation of $f^{-1} \in A$ (when it exists) can be computed by using existing algorithms (e.g., QuasiRecip described in [113]) for computing the inverses of elements in $\tilde{\mathbb{K}}[\boldsymbol{y}]/\operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{K}}}$.

Zero-test over A. A polynomial in $A[z]$ is 0 if and only if all its coefficients are 0 in $A \cong \tilde{\mathbb{K}}[\boldsymbol{y}]/\langle\mathcal{S}\rangle_{\tilde{\mathbb{K}}}$. The latter can be checked via Gröbner basis computation or by using Proposition 2.3.1 (a).

Arithmetic over A. For any polynomials in $A[z]$ represented by $F, G \in \tilde{\mathbb{K}}[\boldsymbol{y}][z]$, $F \pm G$ and FG can be computed in $\tilde{\mathbb{K}}[\boldsymbol{y}][z]$ and then reduced modulo $\operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{K}}}$.

GCD over A. The computation of gcd over A may be achieved by using pgcd (Algorithm 6). For any polynomial set in $A[z]$ represented by $\mathcal{F}$, let

$$\{(F_1, \mathcal{S}_1), \ldots, (F_k, \mathcal{S}_k)\} := \mathsf{pgcd}(\mathcal{F}, \mathcal{S}).$$

What pgcd does is to split the unmixed product A represented by $\mathcal{S}$ into sub-products $A_1, \ldots, A_k$, which are unmixed, disjoint, and represented by $\mathcal{S}_1, \ldots, \mathcal{S}_k$. The polynomials F_i over A_i are returned as the "gcd" of $\mathcal{F}$ over A in the sense that the gcd of $\mathcal{F}$ over each field $\mathbb{K}_{ij}$ in the product A_i is the projection of F_i to $\mathbb{K}_{ij}$. In other words, the gcd's of $\mathcal{F}$ over all the fields $\mathbb{K}_{ij}$ share the same form as F_i for any fixed i. This is consistent with the compact representation of simple sets: encoding the information of multiple field extensions into a single simple set.

6.2 Squarefree decomposition over unmixed products

For any polynomial in $A[z]$ represented by $F \in \tilde{\mathbb{K}}[\boldsymbol{y}][z]$ and any field $\mathbb{K}_i$ in the unmixed product A, let $\overline{F}^{(i)}$ denote the image of F in $\mathbb{K}_i[z]$ under the natural homomorphism.

Definition 6.2.1. A polynomial $f \in A[z]$, represented by $F \in \tilde{\mathbb{K}}[\boldsymbol{y}][z]$, is said to be *squarefree* over A if $\mathcal{S} \cup [F]$ is a regular set in $\tilde{\mathbb{K}}[\boldsymbol{y}][z]$ and $\overline{F}^{(i)}$ is squarefree over $\mathbb{K}_i$ for all $i = 1, \ldots, t$. We call

$$\{(\{[F_{j,1}, a_{j,1}], \ldots, [F_{j,r_j}, a_{j,r_j}]\}, \mathcal{S}_j) : j = 1, \ldots, k\}$$

a *squarefree decomposition* of f over A if

(a) each $\mathcal{S}_j$ is a simple set in $\mathbb{K}[\boldsymbol{x}]$ and $\mathrm{sat}(\mathcal{S}) = \bigcap_{j=1}^{k} \mathrm{sat}(\mathcal{S}_j)$;

(b) $\{[\overline{F}_{j,1}^{(i)}, a_{j,1}], \ldots, [\overline{F}_{j,r_j}^{(i)}, a_{j,r_j}]\}$ is a squarefree decomposition of $\overline{F}^{(i)}$ for all $j = 1, \ldots, k$ and $i = 1, \ldots, t$ such that $\mathbb{K}_i$ occurs in the unmixed product represented by $\mathcal{S}_j$.

Although these definitions are given upon A, which is not necessarily a UFD, we actually reduce the issue of squarefreeness to that over each field in A. By Definition 6.2.1 (b), one can show that for each j the polynomials $F_{j,1}, \ldots, F_{j,r_j}$ in the definition are squarefree over $\tilde{\mathbb{K}}[\boldsymbol{y}]/\mathrm{sat}(\mathcal{S}_j)_{\tilde{\mathbb{K}}}$ and $F = \prod_{l=1}^{r_j} F_{j,l}^{a_{j,l}}$ in $(\tilde{\mathbb{K}}[\boldsymbol{y}]/\mathrm{sat}(\mathcal{S}_j)_{\tilde{\mathbb{K}}})[z]$. This is the way we generalize the concept of squarefree decomposition over UFDs to that over unmixed products of field extensions. The idea has been explored initially in [98] for "generalized squarefree decomposition".

6.2.1 Existing algorithms revisited

In view of the tight connection between simple sets and squarefree decomposition over unmixed products of field extensions, we adapt some existing algorithms for simple decomposition [98, 157], whose target is essentially to extract the squarefree parts of polynomials over A, to compute squarefree decompositions over A. We want to mention that the underlying idea for splitting in the proposed algorithms below is the D5 principle (or called dynamic evaluation), which is incorporated in the pgcd algorithm.

The following algorithm is presented for squarefree decomposition over A for the case when the characteristic of the base field $\mathbb{K}$ is 0. It merely combines squarefree decomposition of univariate polynomials over fields of characteristic 0 and the D5 principle with pgcd. We write it here to illustrate how this combination works.

As usual, the operation pop($\mathbb{D}$) means to choose one element and then delete it from $\mathbb{D}$. For any $[B, C, \mathcal{C}, \mathbb{P}, d]$, B and C are two polynomials waiting for later processing, $\mathcal{C}$ is a simple set at the current state of computation, and $\mathbb{P}$ is used to store computed components, with degrees smaller than d, of the squarefree decomposition.

Proof (Algorithm 19). *Termination.* The **while** loop implements a splitting process based on the algorithm pgcd. We can consider adjoining elements to $\mathbb{D}$ as building up a tree. Elements in $\mathbb{D}$ are the nodes whose depth in the tree is indicated by the

Algorithm 19: Squarefree decomposition for $p = 0$ $\mathbb{S} := \mathsf{sqfChar0}(F, \mathcal{S})$

Input: $\mathcal{S}$, a simple set in $\mathbb{K}[\boldsymbol{x}]$ representing an unmixed product A;
F, a polynomial in $\tilde{\mathbb{K}}[\boldsymbol{y}][z]$ representing a polynomial $f \in A[z] \setminus A$

Output: $\mathbb{S}$, a squarefree decomposition of f over A

19.1 $\mathbb{S} := \emptyset$; $\mathbb{D} := \emptyset$;
19.2 **for** $(C_1, \mathcal{C}) \in \mathsf{pgcd}(\{F, F'\}, \mathcal{S})$ **do**
19.3 $\quad B_1 := F/C_1 \mod \mathcal{C}$;
19.4 $\quad \mathbb{D} := \mathbb{D} \cup \{[B_1, C_1, \mathcal{C}, \emptyset, 1]\}$;
19.5 **end**
19.6 **while** $\mathbb{D} \neq \emptyset$ **do**
19.7 $\quad [B_1, C_1, \mathcal{C}, \mathbb{P}, d] := \mathsf{pop}(\mathbb{D})$;
19.8 $\quad$ **if** $\deg(B_1, z) > 0$ **then**
19.9 $\quad\quad$ **for** $(B_2, \mathcal{A}) \in \mathsf{pgcd}(\{B_1, C_1\}, \mathcal{C})$ **do**
19.10 $\quad\quad\quad C_2 := C_1/B_2 \mod \mathcal{A}$;
19.11 $\quad\quad\quad P := B_1/B_2 \mod \mathcal{A}$;
19.12 $\quad\quad\quad$ **if** $\deg(P, z) > 0$ **then** $\mathbb{P} := \mathbb{P} \cup \{[P, d]\}$;
19.13 $\quad\quad\quad \mathbb{D} := \mathbb{D} \cup \{[B_2, C_2, \mathcal{A}, \mathbb{P}, d+1]\}$;
19.14 $\quad\quad$ **end**
19.15 $\quad$ **end**
19.16 $\quad \mathbb{S} := \mathbb{S} \cup \{(\mathbb{P}, \mathcal{C})\}$;
19.17 **end**
19.18 **return** $\mathbb{S}$

parameter d. To prove the termination of the algorithm, it suffices to show that in each iteration of the **while** loop we have $\deg(B_2, z) < \deg(B_1, z)$, so that each path of the tree is finite. Otherwise, $\deg(B_2, z) = \deg(B_1, z)$; this means that $C_1 = 0$ in Line 19.9, which is impossible by its construction (Lines 19.2 and 19.10) and the assumption that $f \in A[z] \setminus A$ (and thus $F' \neq 0$).

Correctness. Since splitting occurs only in Lines 19.2 and 19.9 with the algorithm pgcd, by the properties of pgcd it can be proved that the regular set $\mathcal{C}$ in each element of the output $\mathbb{S}$ is simple and $\mathrm{sat}(\mathcal{S}) = \bigcap_{(\mathbb{P},\mathcal{C})\in\mathbb{S}} \mathrm{sat}(\mathcal{C})$.

Now we look at any complete path of the tree. Suppose that the depth of the path is s and denote the node in the path by $[B(i), C(i), \mathcal{C}(i), \mathbb{P}(i), i]$ for $i = 1, \ldots, s$.

Let $\mathcal{C}(s)$ represent an unmixed product of $\mathbb{K}_1, \ldots, \mathbb{K}_k$. Then for each $j = 1, \ldots, k$, $\overline{F}^{(j)}$, the canonical image of F over the field $\mathbb{K}_j$, may be assumed to have a squarefree decomposition $\overline{F}^{(j)} = \prod_{l=1}^{s-1} F_l{}^l$. Using the properties (mainly (c)) of pgcd, one can check that

$$\overline{B(i)}^{(j)} = F_i F_{i+1} \cdots F_{s-1}, \quad \overline{C(i)}^{(j)} = F_{i+1} F_{i+2}^2 \cdots F_{s-1}^{s-1-i},$$

and thus at the node of depth i,

$$\overline{P}^{(j)} = \overline{B(i)}^{(j)} / \overline{B(i+1)}^{(j)} = F_i \cdots F_{s-1} / F_{i+1} \cdots F_{s-1} = F_i.$$

This squarefree factor, together with its degree, is stored in $\mathbb{P}$ in Line 19.12, and then stored in $\mathbb{S}$ with the corresponding simple set in Line 19.16. This completes the proof. □

Next we turn to the case when $\mathbb{K} = \mathbb{F}_q$ is a finite field. If $\dim(\mathcal{S}) = 0$ (when $\tilde{\mathbb{F}_q}$ is algebraic extension of $\mathbb{F}_q$ and is also perfect), an algorithm (Algorithm 14) has been presented in Section 5.2.2 for squarefree decomposition over A. Indeed, when $\tilde{\mathbb{K}}$ is a perfect field, algorithms for squarefree decomposition already exist, for pth root extraction, an important ingredient in squarefree decomposition over fields of positive characteristics (See Section 5.2.2 for the detail), is feasible over $\tilde{\mathbb{K}}$.

However, if $\dim(S) > 0$, then $\tilde{\mathbb{F}_q}$ is no longer perfect, and thus the technique in Section 5.2.2 is not valid. Furthermore, the method proposed in Section 5.3 for simple decomposition is not applicable to our problem here either (because computation of radical ideals in the method discards the degree information). Next we study squarefree decomposition over A in the case when $\mathbb{K} = \mathbb{F}_q$ is a finite field and $\dim(S) > 0$.

6.2.2 New algorithm

In Section 3.2.3, we have described the algorithmic structure of squarefree decomposition of polynomials over fields of positive characteristic in Proposition 3.2.9, and the construction of the bases of the sets of derivations of finitely generated field extensions and quotient rings has also been studied. Here we focus our attention on the behaviors of the bases under projections.

Let $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ be a zero-dimensional radical ideal such that $\mathfrak{a} = \bigcap_{i=1}^{k} \mathfrak{m}_i$ with $\mathfrak{m}_i$ maximal, and $\mathfrak{m}_i$ and $\mathfrak{m}_j$ coprime for $i \neq j$. It is easy to verify the one-to-one correspondence between $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{a})$ and $\prod_{i=1}^{k} \mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{m}_i)$. Let π_i be the projection from $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{a})$ to $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{m}_i)$ (note that projections from $\mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$ to $\mathbb{K}[\boldsymbol{x}]/\mathfrak{m}_i$ can be induced). Then any derivation $D \in \mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{a})$ induces a derivation $D^{(i)} \in \mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{m}_i)$ by defining

$$D^{(i)}(\pi_i(F)) := \pi_i(G), \quad i = 1, \ldots, k,$$

if $D(F) = G$ for $F, G \in \mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$.

Proposition 6.2.1. *Let $D_1, \ldots, D_l$ be a basis of the $\mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$-module $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{a})$ and $D_1^{(i)}, \ldots, D_l^{(i)}$ be the derivations of $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{m}_i)$ induced respectively by $D_1, \ldots, D_l$ via π_i for $i = 1, \ldots, k$. Then $D_1^{(i)}, \ldots, D_l^{(i)}$ form a basis of $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{m}_i)$.*

Proof. It suffices to prove that for any $i = 1, \ldots, k$ and $D^{(i)} \in \mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{m}_i)$ there exist $P_1^{(i)}, \ldots, P_l^{(i)} \in \mathbb{K}[\boldsymbol{x}]/\mathfrak{m}_i$ such that $D^{(i)} = \sum_{j=1}^{l} P_j^{(i)} D_j^{(i)}$.

Note that there is a unique derivation $D \in \mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{a})$ which induces $D^{(1)}, \ldots, D^{(k)}$ by $\pi_1, \ldots, \pi_k$ respectively. That is, if $D(F) = G$ for any $F, G \in \mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$, then $D^{(i)}(\pi_i(F)) = \pi_i(G)$. Since $D_1, \ldots, D_l$ is a basis of $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{a})$, there exist $P_1, \ldots, P_l \in \mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$ such that $D = \sum_{j=1}^{l} P_j D_j$, and thus $G = D(F) = \sum_{j=1}^{l} P_j D_j(F)$ for any $F, G \in \mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$.

Supposing $D_j(F) = G_j$, we know that $D_j^{(i)}(\pi_i(F)) = \pi_i(G_j)$, and thus

$$\begin{aligned} D^{(i)}(\pi_i(F)) = \pi_i(G) &= \pi_i\Big(\sum\nolimits_{j=1}^{l} P_j G_j\Big) \\ &= \sum_{j=1}^{l} \pi_i(P_j)\pi_i(G_j) = \sum\nolimits_{j=1}^{l} \pi_i(P_j) D_j^{(i)}(\pi_i(F)). \end{aligned}$$

Setting $P_j^{(i)} = \pi_i(P_j)$, we have $D^{(i)} = \sum_{j=1}^{l} P_j^{(i)} D_j^{(i)}$ because of the arbitrariness of F and G in $\mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$. □

Corollary 6.2.2. *Let $D_1, \ldots, D_l$ be a basis of the $\mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$-module $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{a})$ and $\mathbb{P}$ be a partition of $\{\mathfrak{m}_1, \ldots, \mathfrak{m}_k\}$. Then for each $\mathcal{P} \in \mathbb{P}$ the derivations induced by $D_1, \ldots, D_l$ via the projection from $\mathbb{K}[\boldsymbol{x}]/\mathfrak{a}$ to $\mathbb{K}[\boldsymbol{x}]/\tilde{\mathfrak{a}}$ form a basis of $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\tilde{\mathfrak{a}})$, where $\tilde{\mathfrak{a}} = \bigcap_{\mathfrak{m} \in \mathcal{P}} \mathfrak{m}$.*

Using the relationship between the bases of $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{a})$ and $\mathrm{Der}(\mathbb{K}[\boldsymbol{x}]/\mathfrak{m}_i)$ established above, when handling an unmixed product of field extensions, we only need to compute a basis of the derivations of the product once; the bases of the derivations of all the fields in the product and of some sub-products of the fields can be obtained via projection.

The following algorithm is proposed for squarefree decomposition over unmixed products for $p > 0$ and $\dim(\mathcal{S}) > 0$. In this algorithm there is an important part of pth root extraction over A. Let $q \in A[z]$ represented by Q is a pth power in $A[z]$, then there exist $q_0, q_1, \ldots, q_s$ such that $q = \sum_{j=0}^{s} q_j z^{pj}$, and the extraction of the pth root of q reduces to that in A. This is precisely what we have discussed in Section 5.3.2: For $\tilde{\mathbb{F}}_q$ Condition P holds, and thus we can obtain the unique pth root of q in $A[z]$ by using Condition P [62].

The operation $\mathsf{merge}(\{[A_1, a_1], \ldots, [A_r, a_r]\}, \{[B_1, b_1], \ldots, [B_s, b_s]\})$ first computes the union of the two sets and then replaces any $[A_i, a_i]$ and $[B_j, b_j]$ such that $A_i = B_j$ by $[A_i, a_i + b_j]$. As usual, $\mathbb{F}_q$ denotes the finite field of q elements.

Proof (Algorithm 20). *Termination.* Without loss of generality, we assume that when sqfPos is called recursively in Line 20.19, it terminates if $\deg(C_3, z) < \deg(F, z)$. It is easy to observe from Line 20.18 that $\deg(C_3, z) < \deg(F, z)$ indeed. The termination proof for other parts of the algorithm is the same as that for Algorithm 19.

Correctness. We can view Lines 20.8–20.14 as a tree-building process. Choose any specific path of the tree of depth s and denote by $[B(i), C(i), \mathcal{C}(i), \mathbb{P}(i), i]$ the node of depth i in the path for $i = 1, \ldots, s$. Let $\mathcal{C}(s)$ represent an unmixed product of field extensions $\mathbb{K}_1, \ldots, \mathbb{K}_k$. Then for any $j = 1, \ldots, k$, we can write $\overline{F}^{(j)} = Q \prod_{i=1}^{t} P_i^i$ as in Proposition 3.2.9, where $\overline{F}^{(j)}$ is the canonical image of F over the field $\mathbb{K}_j$.

Using the specification of pgcd, one can show that

$$\overline{B(i)}^{(j)} = QP_iP_{i+1}\cdots P_{s-1}, \quad \overline{C(i)}^{(j)} = QP_{i+1}P_{i+2}^2\cdots P_{s-1}^{s-1-i}.$$

Thus at the node of depth i, we have

$$\overline{P}^{(j)} = \overline{B(i)}^{(j)} / \overline{B(i+1)}^{(j)} = QP_i\cdots P_{s-1}/QP_{i+1}\cdots P_{s-1} = P_i,$$

which is stored with its degree in $\mathbb{P}$ in Line 20.13.

Algorithm 20: Squarefree decomposition for $p > 0$ and $\dim(\mathcal{S}) > 0$

$\mathbb{S} := \mathsf{sqfPos}(F, \mathcal{S})$

Input: $\mathcal{S}$, a simple set in $\mathbb{F}_q[\boldsymbol{x}]$ representing an unmixed product A
F, a polynomial in $\tilde{\mathbb{F}}_q[\boldsymbol{y}][z]$ representing a polynomial $f \in A[z] \setminus A$

Output: $\mathbb{S}$, a squarefree decomposition of f over A

20.1 $\mathbb{S} := \emptyset$; $\mathbb{D} := \emptyset$;
20.2 Compute a basis $D_1, \ldots, D_l$ of $\mathrm{Der}(\tilde{\mathbb{F}}_q[\boldsymbol{y}]/\operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{F}}_q})$;
20.3 **for** $(C_1, \mathcal{C}) \in \mathsf{pgcd}(\{F, F', D_1(F), \ldots, D_l(F)\}, \mathcal{S})$ **do**
20.4 $\quad B_1 := F/C_1 \mod \mathcal{C}$;
20.5 $\quad \mathbb{D} := \mathbb{D} \cup \{[B_1, C_1, \mathcal{C}, \emptyset, 1]\}$;
20.6 **end**
20.7 **while** $\mathbb{D} \neq \emptyset$ **do**
20.8 $\quad [B_1, C_1, \mathcal{C}, \mathbb{P}, d] := \mathsf{pop}(\mathbb{D})$;
20.9 $\quad$ **if** $\deg(B_1, z) > 0$ **then**
20.10 $\quad\quad$ **for** $(B_2, \mathcal{A}) \in \mathsf{pgcd}(\{B_1, C_1\}, \mathcal{C})$ **do**
20.11 $\quad\quad\quad C_2 := C_1/B_2 \mod \mathcal{A}$;
20.12 $\quad\quad\quad P := B_1/B_2 \mod \mathcal{A}$;
20.13 $\quad\quad\quad$ **if** $\deg(P, z) > 0$ **then** $\mathbb{P} := \mathbb{P} \cup \{[P, d]\}$;
20.14 $\quad\quad\quad \mathbb{D} := \mathbb{D} \cup \{[B_2, C_2, \mathcal{A}, \mathbb{P}, d+1]\}$;
20.15 $\quad\quad$ **end**
20.16 $\quad$ **else**
20.17 $\quad\quad$ **if** $\deg(C_1, z) > 0$ **then**
20.18 $\quad\quad\quad C_3 :=$ pth root of C_1 in $(\tilde{\mathbb{F}}_q[\boldsymbol{y}]/\operatorname{sat}(\mathcal{C})_{\tilde{\mathbb{F}}_q})[z]$;
20.19 $\quad\quad\quad$ **for** $(\{[A_1, a_1], \ldots, [A_s, a_s]\}, \mathcal{B}) \in \mathsf{sqfPos}(C_3, \mathcal{C})$ **do**
20.20 $\quad\quad\quad\quad \mathbb{S} := \mathbb{S} \cup \{(\mathsf{merge}(\{[A_1, a_1 p], \ldots, [A_s, a_s p]\}, \mathbb{P}), \mathcal{B})\}$;
20.21 $\quad\quad\quad$ **end**
20.22 $\quad\quad$ **else**
20.23 $\quad\quad\quad \mathbb{S} := \mathbb{S} \cup \{(\mathbb{P}, \mathcal{C})\}$;
20.24 $\quad\quad$ **end**
20.25 $\quad$ **end**
20.26 **end**
20.27 **return** $\mathbb{S}$

Furthermore, $\deg(B(s), z) = 0$, so $C(s) = Q$. If $\deg(C(s), z) > 0$, then by Propositions 3.2.9 and 6.2.1, it can be verified that Q is a nontrivial pth power, whose pth root need be extracted for further computation. In this case the algorithm sqfPos is called recursively to compute a squarefree decomposition of $\sqrt[p]{Q}$, which completes the whole process of squarefree decomposition. □

As explained previously, Algorithm 20 is essentially based on the D5 principle which can extend most operations over fields to those over the products of fields. However, the usefulness of this algorithm lies in the fact that pth power identification and pth root extraction, which are non-trivial in the case of positive characteristics, are performed directly over unmixed products of fields without splitting (as discussed in Section 5.1) so that this algorithm may be potentially more efficient than merely adopting the D5 principle.

According to Algorithm 20, one can easily extract the squarefree parts of polynomials modulo the saturated ideals of simple sets. It is thus straightforward to design an algorithm for simple decomposition based on Algorithm 20. The designed algorithm will be free of computation of radicals of positive-dimensional ideals over finite fields, in contrast to the need of such computation in the algorithm proposed in Section 5.3.

6.3 Factorization over unmixed products

Definition 6.3.1. A polynomial $f \in A[z]$, represented by $F \in \tilde{\mathbb{K}}[\boldsymbol{y}][z]$, is said to be *irreducible* if $\mathcal{S} \cup [F]$ is a regular set in $\tilde{\mathbb{K}}[\boldsymbol{y}][z]$ and $\overline{F}^{(i)}$ is irreducible over $\mathbb{K}_i$ for all $i = 1, \ldots, r$. We call $\{([F_{j,1}, \ldots, F_{j,r_j}], \mathcal{S}_j) : j = 1, \ldots, k\}$ an *irreducible factorization* of f if

(a) each $\mathcal{S}_j$ is a simple set in $\mathbb{K}[\boldsymbol{x}]$ and $\mathrm{sat}(\mathcal{S}) = \bigcap_{j=1}^{k} \mathrm{sat}(\mathcal{S}_j)$;

(b) $\overline{F}_{j,1}^{(i)} \cdots \overline{F}_{j,r_j}^{(i)}$ is an irreducible factorization of $\overline{F}^{(i)}$ for all $j = 1, \ldots, k$ and $i = 1, \ldots, t$ such that $\mathbb{K}_i$ occurs in the unmixed product represented by $\mathcal{S}_j$.

There are polynomials over A whose images over different fields in A have different irreducibilities: the simple set $[(x-1)(x^2+x+1)] \subset \mathbb{Q}[x]$ represents an unmixed product of field extensions; the image of $y^2 - x$ over $\mathbb{Q}[x]/\langle x-1 \rangle$ can be factorized as $(y+1)(y-1)$, while that over $\mathbb{Q}[x]/\langle x^2+x+1 \rangle$ is irreducible.

One can easily infer that all the polynomials $F_{j,1}, \ldots, F_{j,r_j}$ in Definition 6.3.1 are irreducible over $\tilde{\mathbb{K}}[\boldsymbol{y}]/\operatorname{sat}(\mathcal{S}_j)_{\tilde{\mathbb{K}}}$ and $F = \prod_{l=1}^{r_j} F_{j,l}$ in $(\tilde{\mathbb{K}}[\boldsymbol{y}]/\operatorname{sat}(\mathcal{S}_j)_{\tilde{\mathbb{K}}})[z]$. This is how factorization over A generalizes ordinary factorization over UFDs.

6.3.1 Irreducibility test

The key strategy for the design of dynamic algorithms in the preceding section is to reduce squarefree decomposition to gcd computation over unmixed products and introduce dynamic splitting through the algorithm `pgcd`. This strategy will be used again to design algorithms for factorization over unmixed products.

Proposition 6.3.1. *Let $\mathcal{S} \subset \mathbb{K}[\boldsymbol{x}]$ be a simple set representing an unmixed product A and f be a squarefree polynomial in $A[z]$ represented by $F \in \tilde{\mathbb{K}}[\boldsymbol{y}][z] \setminus \tilde{\mathbb{K}}[\boldsymbol{y}]$. Then f is irreducible over A if and only if the number of prime components of $\langle \mathcal{S}, F\rangle_{\tilde{\mathbb{K}}}$ in $\tilde{\mathbb{K}}[\boldsymbol{y}][z]$ is equal to that of $\langle \mathcal{S}\rangle_{\tilde{\mathbb{K}}}$ in $\tilde{\mathbb{K}}[\boldsymbol{y}]$.*

Proof. Suppose that $\langle \mathcal{S}\rangle_{\tilde{\mathbb{K}}} = \bigcap_{i=1}^{t} \mathfrak{p}_i$ is a minimal prime decomposition, with $\mathbb{K}_i = \tilde{\mathbb{K}}[\boldsymbol{y}]/\mathfrak{p}_i$ for $i = 1, \ldots, t$.

($\Longrightarrow$) If f is irreducible over A, then by definition $\overline{F}^{(i)}$ is irreducible over each $\mathbb{K}_i$ for $i = 1, \ldots, t$. Since $\tilde{\mathbb{K}}[\boldsymbol{y}][z]/\langle \mathcal{S}, F\rangle_{\tilde{\mathbb{K}}} \cong \mathbb{K}_1[z]/\langle \overline{F}^{(1)}\rangle \times \cdots \times \mathbb{K}_t[z]/\langle \overline{F}^{(t)}\rangle$ with each $\mathbb{K}_i[z]/\langle \overline{F}^{(i)}\rangle$ already being a field, we know that the number of prime components of $\langle \mathcal{S}, F\rangle_{\tilde{\mathbb{K}}}$ in $\tilde{\mathbb{K}}[\boldsymbol{y}][z]$ is also t.

($\Longleftarrow$) This is easily proved by using

$$\tilde{\mathbb{K}}[\boldsymbol{y}][z]/\langle \mathcal{S}, F\rangle_{\tilde{\mathbb{K}}} \cong \mathbb{K}_1[z]/\langle \overline{F}^{(1)}\rangle \times \cdots \times \mathbb{K}_t[z]/\langle \overline{F}^{(t)}\rangle. \qquad \square$$

Instead of giving a simple algorithm for irreducibility test by direct computation of the numbers of prime components according to Proposition 6.3.1, we will incorporate the test into the factorization algorithm.

By random transformation of variables, one can turn the zero-dimensional ideal $\langle \mathcal{S}, F\rangle_{\tilde{\mathbb{K}}}$ into another ideal in shape position as Definition 4.1.1, with large probability of success [13]. One linear transformation usually suffices [160]. In what follows, we assume that $p = 0$, or $p > 0$ is big enough so that linear transformations of variables can effectively bring the ideals in question to ideals in shape position.

Let an ideal $\mathfrak{a} \subset \mathbb{K}[\boldsymbol{x}]$ be turned into another ideal $\tilde{\mathfrak{a}}$ in shape position with its lex Gröbner basis as in (4.1). The univariate polynomial G_1 contains the majority of information about the zeros of $\tilde{\mathfrak{a}}$. In particular, G_1 is also the minimal polynomial of the multiplication matrix w.r.t. x_1 in the FGLM algorithm [53], which provides us with an effective method to compute G_1 by using the Wiedemann algorithm [55].

Proposition 6.3.2. *Let $\mathfrak{a}$ be a zero-dimensional radical ideal in $\mathbb{K}[\boldsymbol{x}]$. Then every associated prime $\mathfrak{p}$ of $\mathfrak{a}$ remains prime after a linear transformation of variables.*

Proof. It is straightforward and omitted. □

Let $f \in A[z]$ be represented by $F \in \tilde{\mathbb{K}}[\boldsymbol{y}][z]$ and assume that f is squarefree over A. Write F in the form $F = Iz^d + F_{<d} + C$, where $I, C \in \tilde{\mathbb{K}}[\boldsymbol{y}]$ and $0 < \deg(T, z) < d = \deg(F, z)$ for any term T in $F_{<d}$. In order to address irreducibility test and factorization together, we introduce an additional variable λ to construct $\tilde{F} = \lambda I z^d + F_{<d} + C \in \tilde{\mathbb{K}}(\lambda)[\boldsymbol{y}]$, which represents a polynomial $\tilde{f} \in A(t)[z]$.

Proposition 6.3.3. *The polynomial $\tilde{f}$ above is irreducible over $A(t)$ if and only if C is regular modulo* $\mathrm{sat}(\mathcal{S})_{\tilde{\mathbb{K}}}$.

Proof. As f is squarefree over A, $\mathcal{S} \cup [F]$ is a simple set in $\tilde{\mathbb{K}}[\boldsymbol{y}]$. Hence the initial I of F is regular modulo $\mathrm{sat}(\mathcal{S})_{\tilde{\mathbb{K}}}$ and thus $\overline{I}^{(i)} \neq 0$ for all $i = 1, \ldots, t$. Therefore, $\tilde{F}^{(i)}$, the image of $\tilde{F}$ in $\mathbb{K}_i(\lambda)[z]$ by the projection π_i, is equal to $\lambda \overline{I}^{(i)} z^d + \overline{F}^{(i)}_{<d} + \overline{C}^{(i)}$. It can be easily proved that $\tilde{F}^{(i)}$ is reducible over $\mathbb{K}_i(\lambda)$ if and only if $\overline{C}^{(i)} = 0$. By Definition 6.3.1, $\tilde{f}$ is irreducible over $A(\lambda)$ if and only if $\overline{C}^{(i)} \neq 0$ for all $i = 1, \ldots, t$.□

To factorize f or test its irreducibility over A, we assume that C is regular modulo $\mathrm{sat}(\mathcal{S})_{\tilde{\mathbb{K}}}$, so that $\tilde{f}$ is irreducible over $A(\lambda)$. If C is not regular, then $\mathcal{S}$ can be split into $\mathcal{S}_1$ and $\mathcal{S}_2$ as in pgcd such that C is regular modulo $\mathrm{sat}(\mathcal{S}_1)_{\tilde{\mathbb{K}}}$ and is 0 modulo $\mathrm{sat}(\mathcal{S}_2)_{\tilde{\mathbb{K}}}$. Then $\mathcal{S}_1$ may be taken instead of $\mathcal{S}$. Meanwhile, we need to deal with the unmixed product represented by $\mathcal{S}_2$ and the polynomial represented by $\hat{F} = z^s(\hat{I}z^{\hat{d}} + \hat{F}_{<\hat{d}} + \hat{C})$ for some integer $s \geq 1$. In this case, take $\mathcal{S}_2$ and $\hat{I}z^{\hat{d}} + \hat{F}_{<\hat{d}} + \hat{C}$ instead of $\mathcal{S}$ and F respectively and assume that $\hat{C}$ is regular modulo $\mathrm{sat}(\mathcal{S}_2)_{\tilde{\mathbb{K}}}$. The splitting may continue otherwise but will terminate because $\hat{d} < d$. It is part of the factorization

process and does not lead to additional cost. In what follows, we focus our study on the ideal $\langle \mathcal{S}, \tilde{F} \rangle_{\tilde{\mathbb{K}}} \subset \tilde{\mathbb{K}}(\lambda)[\boldsymbol{y}, z]$.

Assume further that $\langle \mathcal{S}, \tilde{F} \rangle_{\tilde{\mathbb{K}}}$ has been transformed by a linear transformation τ of variables into an ideal $\langle \mathcal{S}, \tilde{F} \rangle_{\tilde{\mathbb{K}}}^{\tau}$ in shape position and let $\langle \mathcal{S}, F \rangle_{\tilde{\mathbb{K}}}^{\tau}$ be the ideals obtained from $\langle \mathcal{S}, F \rangle_{\tilde{\mathbb{K}}}$ by using the same transformation. Let the univariate polynomial in the lex Gröbner basis of $\langle \mathcal{S}, \tilde{F} \rangle_{\tilde{\mathbb{K}}}^{\tau}$ (with $z < \boldsymbol{y}$) be $H \in \tilde{\mathbb{K}}(\lambda)[z]$. Since for all the polynomials involved here only their numerators are of influence on irreducibility and factorization, we assume that they are all in $\tilde{\mathbb{K}}[\lambda, \boldsymbol{y}, z]$.

Proposition 6.3.4. *Under the above assumptions, the ideal $\langle \mathcal{S}, F \rangle_{\tilde{\mathbb{K}}}^{\tau}$ in $\tilde{\mathbb{K}}[\boldsymbol{y}, z]$ is in shape position and $H|_{\lambda=1}$ is the univariate polynomial in its lex Gröbner basis.*

Proof. Let $\mathcal{H} = [H(z), y_1 - H_2(z), \ldots, y_r - H_r(z)]$ be the lex Gröbner basis of $\langle \mathcal{S}, \tilde{F} \rangle_{\tilde{\mathbb{K}}}^{\tau}$ (with $z < \boldsymbol{y}$). As $\mathcal{S} \subset \tilde{\mathbb{K}}[\boldsymbol{y}]$ is free of λ, $\langle \mathcal{S}, F \rangle_{\tilde{\mathbb{K}}}^{\tau}$ can be obtained by evaluating $\langle \mathcal{S}, \tilde{F} \rangle_{\tilde{\mathbb{K}}}^{\tau}$ at $\lambda = 1$. Thus $\mathcal{H}|_{\lambda=1}$ is a basis of $\langle \mathcal{S}, F \rangle_{\tilde{\mathbb{K}}}^{\tau}$. The conclusion follows from the structure of $\mathcal{H}$. □

Assume finally that factorization of multivariate polynomials over $\tilde{\mathbb{K}}$ is feasible. Let $H = H_1 \cdots H_k$ be the irreducible factorization of $H \in \tilde{\mathbb{K}}[\lambda, z]$. From Proposition 6.3.2 we know that each irreducible factor H_i of H corresponds to a field extension in the unmixed product represented by the simple set $\mathcal{S} \cup [\tilde{F}]$. Since $\tilde{f}$ is irreducible, $k = t$.

Evaluating the factorization of H at $\lambda = 1$, we obtain the equality $H|_{\lambda=1} = H_1|_{\lambda=1} \cdots H_k|_{\lambda=1}$ in $\tilde{\mathbb{K}}[z]$. Some $H_i|_{\lambda=1}$ may be reducible and can be further factorized. Using Proposition 6.3.2 and counting the numbers of factors in the factorizations of H and $H|_{\lambda=1}$ respectively, we can check whether $\langle \mathcal{S}, F \rangle_{\tilde{\mathbb{K}}}$ has also t prime components. Therefore, by Proposition 6.3.1 we can test the irreducibility of $f \in A[z]$.

The gain of the tricky use of the auxiliary variable λ above is that the number of prime components of $\langle \mathcal{S} \rangle_{\tilde{\mathbb{K}}}$ is determined on the way of computing $H|_{\lambda=1}$ and its factorization for $\langle \mathcal{S}, F \rangle_{\tilde{\mathbb{K}}}$ (which will be further used).

6.3.2 Factorization algorithm

First we reformulate a key technique for factorization over finitely generated field extensions [160, 146] in the setting of unmixed products of field extensions.

Theorem 6.3.5. *Let $\mathcal{S} \subset \mathbb{K}[\boldsymbol{x}]$ be a simple set representing A, an unmixed product of $\mathbb{K}_1, \ldots, \mathbb{K}_t$, and f be a squarefree polynomial in $A[z]$, represented by $F \in \tilde{\mathbb{K}}[\boldsymbol{y}][z]$. Let $\mathbb{K}_i = \tilde{\mathbb{K}}[\boldsymbol{y}]/\mathfrak{p}_i$ with $\mathfrak{p}_i$ maximal and $\langle \mathfrak{p}_i, F\rangle_{\tilde{\mathbb{K}}} \subset \tilde{\mathbb{K}}[\boldsymbol{y}, z]$ be transformed into an ideal $\mathfrak{a}_i$ in shape position by a linear transformation τ for $i = 1, \ldots, t$. Let P_i be the univariate polynomial in the lex Gröbner basis of $\mathfrak{a}_i$ (with $z < \boldsymbol{y}$) as of the form* (4.1) *and $P_i = Q_{i,1}^{m_{i,1}} \cdots Q_{i,k_i}^{m_{i,k_i}}$ be an irreducible factorization over $\tilde{\mathbb{K}}$. Then*

$$\overline{F}^{(i)} = c_i \prod_{j=1}^{k_i} \gcd\left(\overline{F}^{(i)}, \tau^{-1}(Q_{i,j}^{m_{i,j}})\right) \tag{6.1}$$

is an irreducible factorization of $\overline{F}^{(i)}$ over $\mathbb{K}_i$, where $c_i \in \mathbb{K}_i$.

A straightforward way for factorization over A is to separate the fields in A by prime or irreducible decomposition [36, 159]) and then perform factorization over each finitely generated field extension [160]. However, to keep representations of field extensions and factorizations compact, we prefer to represent those field extensions over which factorizations share the same form by their unmixed product if possible. Thus we may need to combine the resulting factorizations of each polynomial over all the field extensions in A. The number of possible combinations could be large. To avoid some of the combinations, we choose an alternative method which reduces the problem of factorization to that of gcd computation over A.

Recall the factorization $H = H_1 \cdots H_k$ and the equality $H|_{\lambda=1} = H_1|_{\lambda=1} \cdots H_k|_{\lambda=1}$ from Section 6.3.1, which provide sufficient information for factorizing f over A. The method of factorization is sketched as follows with three steps.

(1) Since $\tilde{f}$ is irreducible over $A(\lambda)$, each field $\mathbb{K}_i$ in A corresponds to a field $\mathbb{K}_i(\lambda)$ in $A(\lambda)$. By Proposition 6.3.2 each $\mathbb{K}_i(\lambda)$ corresponds to a factor H_i of H. By using Propositions 6.3.1, it can be proved that $\overline{F}^{(i)}$ is irreducible over $\mathbb{K}_i$ in A corresponding to H_i if and only if $H_i|_{\lambda=1}$ is irreducible over $\tilde{\mathbb{K}}$.

Denote by $\mathcal{I}$ the set of all i such that $H_i|_{\lambda=1}$ is irreducible over $\tilde{\mathbb{K}}$ and let $\mathcal{I} = \bigcup_l \mathcal{I}_l$ be a (noncontractible) decomposition of $\mathcal{I}$ such that the product $A_{\mathcal{I}_l}$ of the fields in $\{\mathbb{K}_i : i \in \mathcal{I}_l\}$ is unmixed for all l. Then f is irreducible over all the unmixed products $A_{\mathcal{I}_l}$ and the simple sets representing $A_{\mathcal{I}_l}$ can be computed by $\mathsf{pgcd}(\{\prod_{i\in\mathcal{I}} \tau^{-1}(H_i|_{\lambda=1}), F\}, \mathcal{S})$, yielding one part of the factorization of f over A.

(2) Let $\mathcal{J} := \{1, \ldots, k\} \setminus \mathcal{I}$. Then $H_j|_{\lambda=1}$ is reducible over $\tilde{\mathbb{K}}$ for all $j \in \mathcal{J}$, so $\overline{F}^{(j)}$ is reducible over $\mathbb{K}_j$ in A. Now for each $j \in \mathcal{J}$, use Theorem 6.3.5 and

pgcd to compute a factorization of $\overline{F}^{(j)}$ over $\mathbb{K}_j$ and an irreducible triangular set $\mathcal{S}_j$ representing $\mathbb{K}_j$. This results in the other part of the factorization of f over A as $\{([F_{j,1}, \ldots, F_{j,k_j}], \mathcal{S}_j) : j \in \mathcal{J}\}$ (which is not necessarily a compact representation).

(3) The remaining task is to combine some of the factorizations over field extensions computed in step (2), if possible, as factorizations over unmixed products in a uniform way. Combining triangular sets has been studied in [7, 37, 91]. We are interested essentially in combining irreducible triangular sets $\mathcal{S}_j \cup [F_{j,l}]$ for $l = 1, \ldots, k_j$ and $j \in \mathcal{J}$ to form simple sets in $\tilde{\mathbb{K}}[\boldsymbol{y}][z]$. The method based on Proposition 1 in [37] (under additional conditions on combined triangular sets) can be used for our purpose. We denote by Combine($\mathbb{S}$) the subalgorithm for combining factorizations in $\mathbb{S}$.

The factorization process explained above is described formally as Algorithm 21, in which Wiedemann($\cdot$) denotes the Wiedemman algorithm for computing the minimal polynomial of a matrix and merge($(A, \mathcal{S}), (B, \mathcal{S})$) returns ([A,B], $\mathcal{S}$).

6.4 Examples and experiments

6.4.1 Squarefree decomposition: $p = 0$

To illustrate how Algorithm 19 works, let us consider $\mathcal{S} = [x^2 + u, (y^2 + u)(y - 2)] \subset \mathbb{Q}[u, x, y]$, a simple set representing an unmixed product A of three field extensions

$$\mathbb{Q}(u)[x, y]/\langle x^2 + u, y + x\rangle, \quad \mathbb{Q}(u)[x, y]/\langle x^2 + u, y - x\rangle, \quad \mathbb{Q}(u)[x, y]/\langle x^2 + u, y - 2\rangle.$$

Let $F_1 = z^2 - 2xz - u$ and $F_2 = z^2 - 4z + y^2$. We want to compute a squarefree decomposition of the polynomial $f \in A[z]$ represented $F = F_1 F_2$.

It can be easily verified that $F_1 = \alpha^2$ over all the three field extensions, $F_2 = \beta^2$ only over $\mathbb{Q}(u)[x, y]/\langle x^2 + u, y - 2\rangle$, and F_2 is irreducible over the two other field extensions, where $\alpha = z - x$ and $\beta = z - 2$. Hence we know that

$$\{[\{[\alpha\beta, 2]\}, \mathcal{S}_1], [\{[F_2, 1], [\alpha, 2]\}, \mathcal{S}_2]\}$$

is a squarefree decomposition of f, where $\mathcal{S}_1 = [x^2 + u, y - 2]$ and $\mathcal{S}_2 = [x^2 + u, y^2 + u]$. The following shows how to compute the squarefree decomposition according to Algorithm 19.

Algorithm 21: Factorization over unmixed products $\mathbb{S} := \mathsf{Factor}(F, \mathcal{S})$

Input: $\mathcal{S}$, a simple set in $\mathbb{K}[\boldsymbol{x}]$ representing an unmixed product A;
F, a polynomial in $\tilde{\mathbb{K}}[\boldsymbol{y}][z]$ representing a squarefree polynomial $f \in A[z] \setminus A$

Output: $\mathbb{S}$, a factorization of f over A

21.1 Write $\tilde{F} = \lambda I z^d + F_{<d} + C$ with a new variable λ;
21.2 Assume that C is regular modulo $\mathrm{sat}(S)_{\tilde{\mathbb{K}}}$ and turn $\mathcal{S} \cup [\tilde{F}]$ into a normal set;
21.3 $D :=$ degree of $\langle \mathcal{S}, \tilde{F} \rangle$;
21.4 **repeat**
21.5 Choose a $\tilde{\mathbb{K}}$-linear transformation τ of $\boldsymbol{y}$ and z;
21.6 Construct the multiplication matrix M_τ of $\langle \mathcal{S}, \tilde{F} \rangle^\tau$ w.r.t. $z < \boldsymbol{y}$;
21.7 $H := \mathsf{Wiedemann}(M_\tau)$;
21.8 **until** $\deg(H, z) = D$;
21.9 Factorize H over $\tilde{\mathbb{K}}(\lambda)$ as $H = H_1 \cdots H_t$;
21.10 **if** *H and $H|_{\lambda=1}$ have the same number of factors* **then**
21.11 **return** $\{([F], \mathcal{S})\}$ // f is irreducible
21.12 **end**
21.13 $\tilde{H} := 1$; $\mathbb{S} := \emptyset$;
21.14 **for** $i = 1, \ldots, t$ **do**
21.15 **if** *$H_i|_{\lambda=1}$ is reducible over $\tilde{\mathbb{K}}$* **then**
21.16 Factorize $H_i|_{\lambda=1}$ over $\tilde{\mathbb{K}}$ to obtain its factors $\mathcal{H}_i$;
21.17 $\mathbb{S}' := \emptyset$;
21.18 **for** $Q \in \mathcal{H}_i$ **do**
21.19 $\mathbb{S}' := \mathsf{merge}(\mathbb{S}', \mathsf{pgcd}(\{\tau^{-1}(Q), F\}, \mathcal{S}))$;
21.20 **end**
21.21 $\mathbb{S} := \mathbb{S} \cup \mathbb{S}'$;
21.22 **else**
21.23 $\tilde{H} := \tilde{H} \cdot H_i|_{\lambda=1}$;
21.24 **end**
21.25 **end**
21.26 $\mathbb{S} := \mathsf{Combine}(\mathbb{S}) \cup \mathsf{pgcd}(\{\tau^{-1}(\tilde{H}), F\}, \mathcal{S})$;
21.27 **return** $\mathbb{S}$

First compute F' and then

$$\mathsf{pgcd}(\{F, F'\}, \mathcal{S}) = [(\alpha\beta, \mathcal{S}_1), (\alpha, \mathcal{S}_2)].$$

Computing B_1 in Line 20.4, we have

$$\mathbb{D} := \{[\alpha\beta, \alpha\beta, \mathcal{S}_1, \emptyset, 1], [\alpha F_2, \alpha, \mathcal{S}_2, \emptyset, 1]\}.$$

Pick up the first element of $\mathbb{D}$. Since $\mathsf{pgcd}(\{\alpha\beta, \alpha\beta\}, \mathcal{S}_1) = (\alpha\beta, \mathcal{S}_1)$, we have $C_2 = 1$ and $P = 1$. In this iteration no element is added to $\mathbb{P}$ and $\Delta = [\alpha\beta, 1, \mathcal{S}_1, \emptyset, 2]$ is added to $\mathbb{D}$.

Continue with $\Delta \in \mathbb{D}$: in this iteration we have $\mathsf{pgcd}(\{\alpha\beta, 1\}, \mathcal{S}_1) = (1, \mathcal{S}_1)$ and thus $B_2 = 1$, $C_2 = 1$, and $P = \alpha\beta$. Hence in this step $\mathbb{P} = \{[\alpha\beta, 2]\}$, and we need to handle the remaining $[1, 1, \mathcal{S}_1, \{[\alpha\beta, 2]\}, 3]$. Clearly the criterion $\deg(B_1, z) > 0$ is not satisfied and we end the computation over the field extension represented by $\mathcal{S}_1$.

Computation over the unmixed product represented by $\mathcal{S}_2$ is similar. At the end we will arrive at the squarefree decomposition of f given above.

6.4.2 Squarefree decomposition $p > 0$ and $\dim(\mathcal{S}) > 0$

Consider the simple set $[x^3 - u, (y^2 - u)(y - x)] \subset \mathcal{R}$ which represents an unmixed product A of two field extensions

$$\mathbb{K}_1 \cong \mathcal{R}/\langle x^3 - u, y - x\rangle, \quad \mathbb{K}_2 \cong \mathcal{R}/\langle x^3 - u, y^2 - u\rangle,$$

where $\mathcal{R} = \mathbb{F}_3(u)[x, y]$. Write $\tilde{\mathbb{F}}_3 := \mathbb{F}_3(u)$ and $\tilde{\mathcal{R}} := \tilde{\mathbb{F}}_3[x, y]$. Let $f \in A[z]$ be represented by $F = F_1F_2$, where $F_1 = z^3 - u$ and $F_2 = z^2 - u$. We show how to compute a squarefree decomposition of f using Algorithm 20.

Compute first a basis $D_1 = (x - y)\frac{\partial}{\partial x}$, $D_2 = (x^3 - xy)\frac{\partial}{\partial x} + (x^3 - y^2)\frac{\partial}{\partial y}$ of $\mathrm{Der}(\tilde{\mathcal{R}}/\operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{F}}_3})$ by using the method described in [68]. With this basis,

$$\mathsf{pgcd}(\{F, F', D_1(F), D_2(F)\}, \mathcal{S}) = (F_1, \mathcal{S})$$

in Line 20.3 and thus $B_1 = F_2$. Continue with the algorithm: $\mathsf{pgcd}(\{F_2, F_1\}, \mathcal{S}) = (1, \mathcal{S})$ in Line 20.10 and consequently $C_2 = F_1$ and $P = F_2$. Then $[F_2, 1]$ is adjoined to $\mathbb{P}$.

Since $\deg(F_1, z) > 0$, F_1 is a third power in $(\tilde{\mathcal{R}}/\operatorname{sat}(\mathcal{S})_{\tilde{\mathbb{F}}_3})[z]$. Its third root $F_3 = z - x$ is computed by using the method described in [62, 118]. Clearly $\{[F_3, 1], \mathcal{S}\}$ is the result in the recursive call in Line 20.19. Therefore, the complete squarefree decomposition of F over A is $\{(\{[F_3, 3], [F_2, 1]\}, \mathcal{S})\}$.

Note that the factorizations of F_2 over $\mathbb{K}_1$ and $\mathbb{K}_2$ are different: they are F_2 and $(z + y)(z - y)$ respectively, but this does not affect the fact that F_2 is squarefree over A. That is why no splitting occurs in any of the pgcd calls in Algorithm 20.

6.4.3 Factorization

To illustrate how Algorithm 21 works, let us consider $\mathcal{S} = [x^2 + u, (y^2 + u)(y - 2)] \subset \mathbb{Q}[u, x, y]$, a simple set representing an unmixed product A of three field extensions. We want to compute an irreducible factorization of $f \in A[z]$ represented by $F = z^2 - (u^2 + 2)z + u^2 y$.

Set $\tilde{F} = \lambda z^2 - (u^2+2)z + u^2 y$. One sees easily that $u^2 y$ is regular modulo $\operatorname{sat}(\mathcal{S})_{\mathbb{Q}(u)}$. As $\mathcal{S} \cup [\tilde{F}]$ is already a normal set, it is also the Gröbner basis of $\langle \mathcal{S}, \tilde{F} \rangle$. The degree of $\langle \mathcal{S}, \tilde{F} \rangle$ is 12. Choose a linear transformation $\tau : z + 2\,x + y \mapsto z$ and construct the multiplication matrix M_τ over $\mathbb{Q}(u, \lambda)$ of $\langle \mathcal{S}, \tilde{F} \rangle^\tau$ w.r.t. the lex ordering determined by $z < x < y$.

The univariate polynomial H in the lex Gröbner basis of $\langle \mathcal{S}, \tilde{F} \rangle^\tau$ computed by the Wiedemann algorithm is of degree 12 in z, which ensures that $\langle \mathcal{S}, \tilde{F} \rangle^\tau$and $\langle \mathcal{S}, F \rangle^\tau$ are both in shape position. Factorizing H over $\mathbb{Q}(u, \lambda)$ and $H|_{\lambda=1}$ over $\mathbb{Q}(u)$ shows that

$$H = H_1 H_2 H_3, \quad H|_{\lambda=1} = H_1|_{\lambda=1}(H_{21} H_{22}) H_3|_{\lambda=1}.$$

Since the numbers of irreducible factors of H and $H|_{\lambda=1}$ are different, f is reducible over A. As $H_1|_{\lambda=1}$ and $H_3|_{\lambda=1}$ are irreducible and $H_2|_{\lambda=1}$ can be factorized as $H_{21} H_{22}$, first compute

$$\mathsf{pgcd}(\{F, \tau^{-1}(H_1|_{\lambda=1} H_3|_{\lambda=1})\}, \mathcal{S}) = (F, \mathcal{S}_1),$$

where $\mathcal{S}_1 = [x^2 + u, (y + x)(y - x)]$. This means that f is irreducible over the unmixed product represented by the simple set $\mathcal{S}_1$. Then compute

$$\mathsf{pgcd}(\{F, \tau^{-1}(H_{21})\}, \mathcal{S}) = (\alpha, \mathcal{S}_2), \quad \mathsf{pgcd}(\{F, \tau^{-1}(H_{22})\}, \mathcal{S}) = (\beta, \mathcal{S}_2),$$

where $\alpha = z - 2$, $\beta = z - u^2$, and $\mathcal{S}_2 = [x^2 + u, y - 2]$.

The output $\{([F], \mathcal{S}_1), ([\alpha, \beta], \mathcal{S}_2)\}$ is an irreducible factorization of f over the unmixed product A represented by $\mathcal{S}$. For this example no combination of factorization occurs, as there is only one j such that $H_j|_{\lambda=1}$ is reducible.

6.4.4 Preliminary experiments

We have implemented Algorithms 20 and 21 (mainly) in Maple 14. The implementation makes use of the RegularChains library for computations with regular sets [93] and calls external functions from Singular [41] for Algorithm 20. We have made preliminary experiments with the implementation on an Intel(R) Pentium(R) P8700 CPU 2.53 GHz with 1.89 G RAM under Windows XP Professional SP3.

Experiments for Algorithm 20 have been carried out with examples derived from those used in [118]. The dominant part of computation is the pth root extraction when it is necessary. For example, computing $\mathsf{sqfPos}((z^{53}-s)(z^{53}-t), [(x^{53}-s)(x^{53}-t)])$ over $\mathbb{F}_{53}$ takes about 130.1 CPU seconds, of which 129.5 seconds are spent for computing all the pth roots. During the pth root extraction for this example, linear systems with coefficient matrices of size 2809×53 have to be constructed and solved. It may be concluded that the larger the characteristic p is, the more difficult the computation for the pth root extraction becomes.

For Algorithm 21, our experiments on several constructed examples show that dominant computations take place in pgcd in Lines 21.19 and 21.26, and the efficiency of the algorithm depends largely on the degree of $\langle \mathcal{S}, \tilde{F} \rangle$ and the number of parameters. For example, the factorization $\mathsf{Factor}(z^{2i} - 2\,u, [(x^2 - u)(x - u), (y^2 - 2)(y - u)])$ over $\mathbb{Q}$ for $i = 1, \ldots, 6$ takes about 2, 15, 202, 576, 2426, and more than 3600 CPU seconds respectively. For $i = 5$, the time for computing pgcd is about 2163 seconds in total and the polynomial $\tau^{-1}(\tilde{H})$ computed in Line 21.26 contains 1244 terms and is of degree 50 in z.

Concluding remarks and future work

Based on the useful complexity analyses on standard operations like GCD over unmixed products of field extensions [38], the computational complexity of Algorithms

20 may be analyzed. This is one direction of our future work. Furthermore, our implementations of the proposed algorithms need further refinement and improvement, with more experiments. In particular, the proposed algorithms should be compared with the algorithms adapted from those over field extensions by the D5 principle.

Chapter 7

Applications

The results in this chapter are based on the joint work with Xiaoliang Li, Wei Niu, and Dongming Wang (Section 7.1, published in [97]) and with Jean-Charles Faugère (Section 7.2, on-going work).

7.1 Detection of steady states and their numbers for finite biological models

7.1.1 Introduction

Continuous and discrete dynamical systems are widely used for the modeling of biological phenomena. A general algebraic approach has been proposed in [120, 161] to study real equilibria, their stability, bifurcations and limit cycles of biological networks modeled as continuous dynamical systems. Discrete dynamical systems are simply structured and intuitive to biologists. In particular, discrete systems can be modeled with a small amount of data.

For discrete dynamical systems, the time domain is on fixed discrete intervals (not the real axis $\mathbb{R}$). A finite biological model, usually used to describe finite dynamical systems, is one kind of discrete biological models where the variables and parameters are values from a finite field $\mathbb{F}_q$. As in the continuous case, it is difficult to find the analytical solution of a finite biological model (if such solutions exist at all), so the detection of their steady states and the numbers becomes important to study the

qualitative behaviors of their solutions, and thus their dynamical characteristics.

We only consider systems of first-order autonomous discrete difference equations of the form

$$\begin{cases} x_1(t+1) = \phi_1(u_1, \ldots, u_m, x_1(t), \ldots, x_n(t)), \\ \qquad\qquad \vdots \\ x_n(t+1) = \phi_n(u_1, \ldots, u_m, x_1(t), \ldots, x_n(t)), \end{cases} \tag{7.1}$$

where $u_1, \ldots, u_m$ are parameters independent of t, $x_1, \ldots, x_n$ are variables, and $\phi_i : {\mathbb{F}_q}^{m+n} \to \mathbb{F}_q$ is a map for $i = 1, \ldots, n$ with $\mathbb{F}_q$ a finite field. Let $\boldsymbol{u} = (u_1, \ldots, u_m)$, $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\Phi = (\phi_1, \ldots, \phi_n)$.

A point $\overline{\boldsymbol{x}} \in {\mathbb{F}_q}^n$ is said to be a *steady state* of system (7.1) if $\overline{\boldsymbol{x}} = \phi(\overline{\boldsymbol{u}}, \overline{\boldsymbol{x}})$. By this definition one sees that after the parameter $\overline{\boldsymbol{u}}$ is fixed, the point $\overline{\boldsymbol{x}}$ keeps unchanged while t increases. The purpose of this part is to study the steady states and their numbers of finite biological systems which can be modeled as (7.1).

Our attention is focused on the simplest but most widely used finite dynamical systems, Boolean networks, where ϕ_i is written in terms of the Boolean operators $\vee$, $\wedge$, $\neg$; values of state variables $\boldsymbol{x}$ and parameters $\boldsymbol{u}$ are taken from $\mathbb{F}_2^n$ and $\mathbb{F}_2^m$ respectively. Boolean networks can describe qualitatively the structures and dynamical characteristics of biological systems, and are usually used to model complicated systems like gene regulatory networks [145]. For such systems, most of the existing studies focus on random Boolean networks [81]. Recently, methods based on Gröbner bases [87, 88] and SAT algorithms [148] are applied to detecting equilibria for given Boolean networks.

7.1.2 Detecting steady states and their numbers

Boolean functions can be translated into polynomial functions by using the rules $x \wedge y = xy$, $x \vee y = x + y + xy$ and $\neg x = x + 1$ (see, e.g., [88]). Then determining the number of steady states of a Boolean network can be reduced to counting the number of solutions of the equation system for $\boldsymbol{x}$ in $\mathbb{F}_2^n$

$$F_1(\boldsymbol{u}, \boldsymbol{x}) - x_1 = 0, \ldots, F_n(\boldsymbol{u}, \boldsymbol{x}) - x_n = 0, \tag{7.2}$$

where F_i is the polynomial transformed from ϕ_i for $i = 1, \ldots, n$.

We are mainly interested in the positive-dimensional case, that is system (7.2) is a parametric one. It should be emphasized that the solutions of our interest are in $\mathbb{F}_2^n$,

instead of $\overline{\mathbb{F}}_2^n$. Therefore to restrict the solutions to $\mathbb{F}_2^n$, we adjoin the *field equations* $x_i^2 + x_i = 0$ $(i = 1, \ldots, n)$ to the original polynomial equations $F_1 = 0, \ldots, F_n = 0$ to have a new equation set

$$F_1 = \cdots = F_n = x_1^2 + x_1 = \cdots = x_n^2 + x_n = 0. \tag{7.3}$$

It can be easily shown the solutions of (7.3) are precisely all the solutions of $F_1 = \cdots = F_n = 0$ in $\mathbb{F}_2^n$.

To solve such polynomial systems over finite fields, we mainly adopt two kinds of method based on Gröbner bases and triangular sets respectively: the traversal method which traverses all the possible parameters and solves the resulting parameter-free systems with Gröbner bases, and the method based triangular decomposition for Boolean systems [64]. All the experiments in this part were made in MAGMA 2.16-1, running under Scientific Linux OS release 5.4 on 8 Xeon(R) CPUs E5420 2.50 GHz with 20.55 G RAM.

Traversal method based on Gröbner bases

Here the traversal method for a parametric system means to traverse all the parameter specifications and solve the parameter-free system directly for each specification with Gröbner bases. This method is feasible because there are only finitely many points in the parameter space and it is efficient when the number of parameters is relatively small. The Gröbner basis of a Boolean system has a quasi-triangular form which makes it easy to count the number of solutions of the system.

As an example for the traversal method, we consider a Boolean network of the segment polarity genes in Drosophila melanogaster studied in [2, 87]. As shown in Figure 7.1, each cell consists of 15 nodes which are abstracted from mRNA and proteins inside, and every cell interacts with its neighbor cells with 6 nodes. To some extent, four successive cells are regarded as a unit, for a given gene is expressed in every four cells when expression of the segment polarity genes begins.

We study the two cases of 4 and 8 cells with nodes inside these cells considered as variables and 6 nodes (from the neighbors) that influence these cells as parameters. The Boolean functions that govern the evolution process can be derived from Figure 7.1, similarly as in [87]. Hence these Boolean systems have 60 and 120 variables

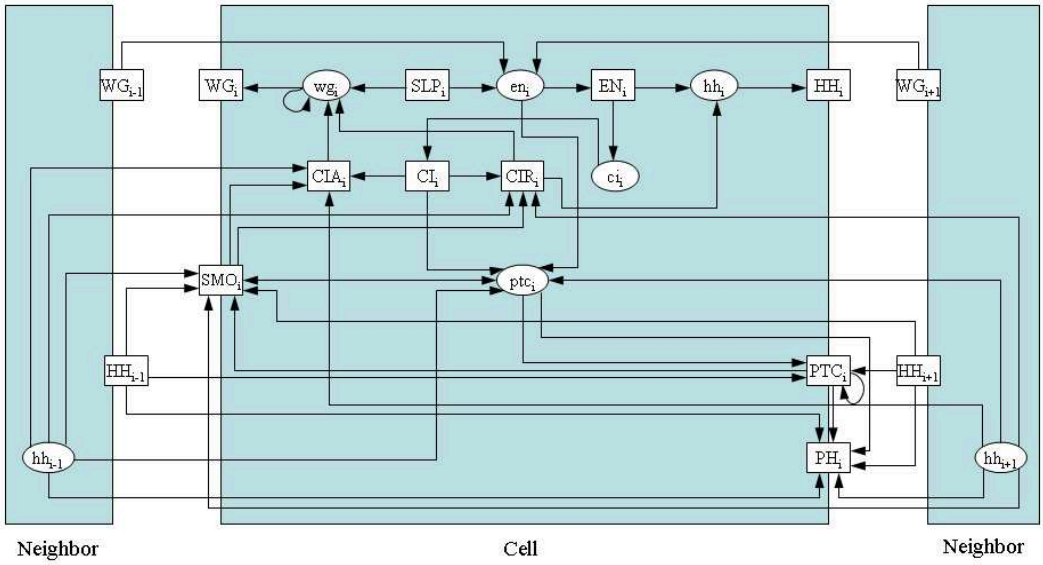

Figure 7.1: Graph of interactions in the Boolean network.

respectively, both with 6 parameters. Table 7.1 illustrates the solution number for each specification of the 6 parameters for both instances. In this table, the parameters in the list correspond to $\mathrm{WG}_0, \mathrm{HH}_0, \mathrm{hh}_0, \mathrm{WG}_{m+1}, \mathrm{hh}_{m+1}$ and HH_{m+1} as in Figure 7.1 respectively, where m is the cell index.

All the computations for the 4-cell and 8-cell cases, including the solution counting, took 1.94 and 38.62 CPU seconds respectively.

As seen from the experimental performances, the traversal method based on Gröbner bases has the advantage of high efficiency for Boolean polynomial systems derived from biological models. Such systems usually have a simpler structure (for example, several nodes in Figure 7.1 are influenced only by one other node), compared with random Boolean polynomial systems which are relatively hard to solve.

Triangular set method

The field $\mathbb{F}_2$ is a special finite field where specialized techniques can be applied. An algorithm for computing triangular decomposition from Boolean polynomial sets has been proposed [64].

Given a polynomial set $\mathcal{F} \subseteq \mathbb{F}_2[\boldsymbol{x}]$, denote by $\mathrm{Zero}_2(\mathcal{F})$ the set of common zeros of $\mathcal{F}$ in $\mathbb{F}_2^n$. One can compute triangular sets $\mathcal{T}_i = [T_{i1}, \ldots, T_{ir_i}]$ $(i = 1, \ldots, r)$ from $\mathcal{F}$ such that

Table 7.1: Numbers of steady states for the 4-cell/8-cell instances

Specification	N_1/N_2	Specification	N_1/N_2	Specification	N_1/N_2
0, 0, 0, 0, 0, 0	7/49	0, 1, 1, 0, 1, 0	6/42	0, 0, 1, 1, 0, 1	0/0
1, 0, 0, 0, 0, 0	5/35	1, 1, 1, 0, 1, 0	3/21	1, 0, 1, 1, 0, 1	0/0
0, 1, 0, 0, 0, 0	5/35	0, 0, 0, 1, 1, 0	7/49	0, 1, 1, 1, 0, 1	0/0
1, 1, 0, 0, 0, 0	3/21	1, 0, 0, 1, 1, 0	5/35	1, 1, 1, 1, 0, 1	0/0
0, 0, 1, 0, 0, 0	7/49	0, 1, 0, 1, 1, 0	5/35	0, 0, 0, 0, 1, 1	7/49
1, 0, 1, 0, 0, 0	5/35	1, 1, 0, 1, 1, 0	3/21	1, 0, 0, 0, 1, 1	5/35
0, 1, 1, 0, 0, 0	6/42	0, 0, 1, 1, 1, 0	7/49	0, 1, 0, 0, 1, 1	5/35
1, 1, 1, 0, 0, 0	3/21	1, 0, 1, 1, 1, 0	5/35	1, 1, 0, 0, 1, 1	3/21
0, 0, 0, 1, 0, 0	7/49	0, 1, 1, 1, 1, 0	6/42	0, 0, 1, 0, 1, 1	7/49
1, 0, 0, 1, 0, 0	5/35	1, 1, 1, 1, 1, 0	3/21	1, 0, 1, 0, 1, 1	5/35
0, 1, 0, 1, 0, 0	5/35	0, 0, 0, 0, 0, 1	0/0	0, 1, 1, 0, 1, 1	6/42
1, 1, 0, 1, 0, 0	3/21	1, 0, 0, 0, 0, 1	0/0	1, 1, 1, 0, 1, 1	3/21
0, 0, 1, 1, 0, 0	7/49	0, 1, 0, 0, 0, 1	0/0	0, 0, 0, 1, 1, 1	7/49
1, 0, 1, 1, 0, 0	5/35	1, 1, 0, 0, 0, 1	0/0	1, 0, 0, 1, 1, 1	5/35
0, 1, 1, 1, 0, 0	6/42	0, 0, 1, 0, 0, 1	0/0	0, 1, 0, 1, 1, 1	5/35
1, 1, 1, 1, 0, 0	3/21	1, 0, 1, 0, 0, 1	0/0	1, 1, 0, 1, 1, 1	3/21
0, 0, 0, 0, 1, 0	7/49	0, 1, 1, 0, 0, 1	0/0	0, 0, 1, 1, 1, 1	7/49
1, 0, 0, 0, 1, 0	5/35	1, 1, 1, 0, 0, 1	0/0	1, 0, 1, 1, 1, 1	5/35
0, 1, 0, 0, 1, 0	5/35	0, 0, 0, 1, 0, 1	0/0	0, 1, 1, 1, 1, 1	6/42
1, 1, 0, 0, 1, 0	3/21	1, 0, 0, 1, 0, 1	0/0	1, 1, 1, 1, 1, 1	3/21
0, 0, 1, 0, 1, 0	7/49	0, 1, 0, 1, 0, 1	0/0		
1, 0, 1, 0, 1, 0	5/35	1, 1, 0, 1, 0, 1	0/0		

(a) $\mathrm{Zero}_2(\mathcal{F}) = \bigcup_{i=1}^{r} \mathrm{Zero}_2(\mathcal{T}_i)$, and $\mathrm{Zero}_2(\mathcal{T}_1), \ldots, \mathrm{Zero}_2(\mathcal{T}_r)$ are pairwisely disjoint;

(b) T_{ij} is of the form $x_{c_i} + U_j$, where $U_j \in \mathbb{F}_2[x_1, \ldots, x_{c_i-1}]$.

It is obvious by Item (b) above that the number $|\,\mathrm{Zero}_2(\mathcal{T}_i)|$ of common zeros of $\mathcal{T}_i$ is 2^{d_i}, where d_i is the number of parameters of $\mathcal{T}_i$, and thus $|\,\mathrm{Zero}_2(\mathcal{P})| = \sum_{i=1}^{r} 2^{d_i}$. Then with this special triangular decomposition we can not only obtain all the solutions, but we can also directly count the number of solutions without explicit computing them. This is especially useful if only the number of solutions are of the main interest.

When handling parametric polynomial systems with both explicit parameters $\boldsymbol{u}$ and variables $\boldsymbol{x}$, we first regard all of them as variables, but with the ordering $\boldsymbol{u} < \boldsymbol{x}$, that is all the parameters are ordered smaller than the variables. Then according to the definition of triangular sets, each triangular set $\mathcal{T}_i$ in the decomposition can be further written as $\mathcal{T}_i = [\mathcal{T}_i^{(1)}, \mathcal{T}_i^{(2)}]$, where $\mathcal{T}_i^{(1)}$ and $\mathcal{T}_i^{(2)}$ are triangular sets in

$\mathbb{F}_2[\boldsymbol{u}]$ and $\mathbb{F}_2[\boldsymbol{u}][\boldsymbol{x}]$ respectively. Note that all $\mathcal{T}_i^{(1)}$ $(i = 1, \ldots, r)$ indeed describe all the possible parameter values for the system to have a solution. With the same formula for counting the solutions as above, we can compute the number of solutions $|\operatorname{Zero}_2(\mathcal{T}_i^{(2)})|$ for the parameter values determined by $\mathcal{T}_i^{(1)}$. In particular, this number of solutions is consistent for all the values determined by $\mathcal{T}_i^{(1)}$.

7.2 Sparse FGLM algorithm for interpolation problem in list decoding

7.2.1 Introduction

In Coding Theory, list decoding is a useful method for decoding error-correcting codes of large error rates [48, 105]. Guruswami and Sudan developed an efficient list decoding algorithm for Reed-Solomon codes, which is able to correct more errors than traditionary decoding methods [70]. Following this result Koetter and Vardy proposed a soft-decision list decoding algorithm for Reed-Solomon [84]. The list decoding forms a part of study on Algebraic Geometric codes [72].

The list decoding algorithm proposed in [70, 84] consists of two steps: the interpolation step and the factorization step. We are interested in the interpolation step, where the main purpose is to find a bivariate polynomial which has the minimal weighted degree and passes through a number of points with given multiplicities. This polynomial can be easily computed via solving a linear system constructed from the given constrains, if efficiency is not the main concern. Hence the existence of this polynomial is already guaranteed, and the study on this interpolation step in the list decoding algorithm mainly focuses on finding the target polynomial in an efficient way.

There have been many effective and efficient algorithms proposed for solving this interpolation problem from different perspectives [119, 125, 92, 151]. In particular, in some algorithms the interpolation problem has already been formulated as finding the minimal polynomial w.r.t. the given weighted term ordering of a polynomial ideal defined by the given points and multiplicities [92]. In the follwing we further reduce the problem to that of change of orderings of Gröbner bases of this ideal, which may be solved with our algorithms proposed in Chapter 4.

7.2.2 From interpolation to change of orderings

The interpolation problem in the list decoding algorithm is: given points $(x_1, y_1), \ldots, (x_k, y_k) \in \mathbb{F}_q \times \mathbb{F}_q$ with corresponding integers $m_1, \ldots, m_k$ representing their multiplicities and a term ordering $<$ (usually a weighed degree ordering because of the Coding Theory background), find a polynomial in $\mathbb{F}_q[x, y]$ which passes through these points with at least the given multiplicities and is minimal w.r.t. $<$.

For any polynomial $P \in \mathbb{F}_q[x, y]$ and a point (x_i, y_i) with the multiplicity m_i $(1 \leq i \leq k)$, we can write

$$P(x, y) = \sum_{r,s} a_{i,r,s}(x - x_i)^r (y - y_i)^s.$$

It is clear that a necessary and sufficient condition for (x_i, y_i) to be a zero of P with multiplicities at least m_i is that $a_{i,r,s} = 0$ for any r and s such that $r + s < m_i$. In the terminology of polynomial ideals, one sees that such a polynomial P is in the ideal

$$\mathfrak{a}_i := \langle (x - x_i)^r (y - y_i)^{m_i - r} : r = 0, \ldots, m_i \rangle.$$

With all points (x_i, y_i) and their multiplicities m_i taken into consideration, the target polynomial Q in the interpolation problem is indeed the minimal polynomial in the ideal $\mathfrak{a} = \bigcap_{i=i}^{k} \mathfrak{a}_i$ w.r.t. $<$. That is also the minimal polynomial in the Gröbner basis of $\mathfrak{a}$ w.r.t. $<$.

Next we turn our interest to the quotient rings $\mathbb{F}_q[x, y]/\mathfrak{a}$ and $\mathbb{F}_q[x, y]/\mathfrak{a}_i$. First it is clear that for each $i = 1, \ldots, k$, the generators $\{(x - x_i)^r (y - y_i)^{m_i - r} : r = 0, \ldots, m_i\}$ of $\mathfrak{a}_i$ already form a Gröbner basis of $\mathfrak{a}_i$ w.r.t. the DRL ordering. Therefore we know that $\mathfrak{a}_i$ is zero-dimensional, and a basis of $\mathbb{F}_q[x, y]/\mathfrak{a}_i$ is $B_i := \{x^i y^j : i + j < m_i\}$. With the above information, one can construct the multiplications matrices of $\mathfrak{a}_i$ in the FGLM algorithm for x and y w.r.t. the DRL ordering, denoted by $T_{i,x}$ and $T_{i,y}$ respectively.

Since for all $i \neq j$, $\mathsf{V}(\mathfrak{a}_i + \mathfrak{a}_j) = \mathsf{V}(\mathfrak{a}_i) \cap \mathsf{V}(\mathfrak{a}_j) = \emptyset$, one knows that $\mathfrak{a}_i + \mathfrak{a}_j = \mathbb{F}_q[x, y]$. Then by the Chinese Remainder Theorem (Theorem 3.1.13), one has

$$\mathbb{F}_q[x, y]/\mathfrak{a} \cong \prod_{i=1}^{k} \mathbb{F}_q[x, y]/\mathfrak{a}_i.$$

The basis of the vector space $\mathbb{F}_q[x, y]/\mathfrak{a}_i$ being B_i, one can construct the basis B of the direct product $\mathbb{F}_q[x, y]/\mathfrak{a}$ of $\mathbb{F}_q[x, y]/\mathfrak{a}_1, \ldots, \mathbb{F}_q[x, y]/\mathfrak{a}_k$ componentwise. In particular,

the multiplication matrices T_x and T_y of $\mathfrak{a}$ for x and y respectively w.r.t the DRL ordering can be obtained by adjoining $T_{1,x}, \ldots, T_{k,x}$ and $T_{1,y}, \ldots, T_{k,y}$ diagonally.

At this point we have a clear understanding of the structure of $\mathbb{F}_q[x,y]/\mathfrak{a}$, which is revealed by studying its relationship with $\mathbb{F}_q[x,y]/\mathfrak{a}_1, \ldots, \mathbb{F}_q[x,y]/\mathfrak{a}_k$. As analyzed previously, the polynomial Q we want to compute for the interpolation problem is the minimal polynomial of $\mathfrak{a}$ w.r.t. $<$. With the Gröbner basis and multiplication matrices of $\mathfrak{a}$ w.r.t. DRL known, the problem of finding Q is reduced to changing the ordering of its Gröbner basis from DRL to $<$.

7.2.3 Preliminary results

Since the problem of finding the minimal polynomial has been reduced to changing the orderings of Gröbner bases of $\mathfrak{a}$, efficient algorithms for this purpose proposed in Chapter 4 may become applicable. The preliminary observations obtained from our applications of these algorithms to this interpolation problem are summarized as follows.

The basic result is that the ideal $\mathfrak{a}$ is not in shape position, therefore we have to apply the BMS-based algorithm for the general case to $\mathfrak{a}$. However, even this algorithm fails to return the correct Gröbner basis of $\mathfrak{a}$ w.r.t. $<$ (remember that it can be checked whether we obtained the correct result). Since the BMS-based algorithm works for most zero-dimensional ideals (for example randomly generated ones), we can conclude that the ideal $\mathfrak{a}$ bears a very special structure, which is also clear from the way it is constructed. We are still working on how to adapt the BMS-based algorithm to this specific ideal. One potential technique needed here is the generalization of Wiedemann algorithm to the multivariate case

One advantages of applying the BMS-based algorithm to find the minimal polynomial in the interpolation polynomial in the list decoding algorithm is that the algebraic structure behind the problem is made clearer (as discussed above), and the process of finding the minimal polynomial as the change of orderings of Gröbner bases is also straightforward, and hopefully efficient.

Another potential advantage is related to the complexity result, both for construction of T_x and T_y and for computation as in Theorem 4.2.5, may be clearly analyzed. The construction process of all the multiplication matrices $T_{i,x}$ and $T_{i,y}$ for $\mathfrak{a}_i$ are

clear for $i = 1, \ldots, k$. This makes it possible to directly analyze the cost to construct T_x and T_y and their sparsity, which is a key factor in the complexity analysis for the BMS-based algorithm for change of orderings.

On the other hand, one disadvantage is that though only the minimal polynomial is needed (it can also be checked with similar techniques whether the corrected one is computed) and the algorithm will be stopped once this polynomial is computed, we will still have computed a large majority of the Gröbner basis with the BMS-based algorithm up to the termination. The underlying reason for this phenomenon is that the target term ordering is a (weighted) degree one, and the degree of the minimal polynomial we want to compute is usually close to those of other polynomials in the Gröbner basis w.r.t. this term ordering. And therefore with the BMS-based algorithm which computes the minimal generating set degree by degree, we cannot simply compute the target minimal polynomial without the others in the Gröbner basis.

Conclusions

In the introduction relatively detailed descriptions of the contributions and perspectives of this thesis have been presented. We end this thesis by briefly summarizing and emphasizing the original scientific contributions.

(a) Efficient algorithms for change of orderings of Gröbner bases of zero-dimensional ideals are proposed by using the sparsity of multiplication matrices, and the computational complexities of these algorithms and the sparsity of one multiplication matrix for generic polynomial systems are analyzed.

(b) Algorithms for decomposing both zero-dimensional and positive-dimensional polynomial sets over finite fields into simple sets are proposed and implemented.

(c) The concepts of squarefreeness and irreducibility of polynomials over unmixed products of field extensions are introduced, and algorithms for squarefree decomposition and factorization over such unmixed products are designed.

(d) Methods for solving polynomial systems over finite fields are applied to detect the steady states of finite biological systems and their numbers, and the method to find the minimal polynomial in the interpolation problem in Sudan's list decoding algorithm with the proposed algorithm for change of ordering is discussed.

Bibliography

[1] E. Aguirre, A. Jarrah, and R. Laubenbacher. Generic ideals and Moreno-Socías conjucture. In *Proceedings of ISSAC 2001*, pages 21–23. ACM Press, 2001.

[2] R. Albert and H. G. Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster*. *Journal of Theoretical Biology*, 223(1):1–18, 2003.

[3] G. Ars, J.-C. Faugère, H. Imai, M. Kawazoe, and M. Sugita. Comparison between XL and Gröbner basis algorithms. *Advances in Cryptology–ASIACRYPT 2004*, pages 157–167, 2004.

[4] M. Atiyah and I. MacDonald. *Introduction to Commutative Algebra*. Westview Press/Addison-Wesley, Reading, MA, 1969.

[5] P. Aubry, D. Lazard, and M. Moreno Maza. On the theories of triangular sets. *Journal of Symbolic Computation*, 28(1–2):105–124, 1999.

[6] P. Aubry and M. Moreno Maza. Triangular sets for solving polynomial systems: A comparative implementation of four methods. *Journal of Symbolic Computation*, 28(1–2):125–154, 1999.

[7] P. Aubry and A. Valibouze. Using Galois ideals for computing relative resolvents. *Journal of Symbolic Computation*, 30(6):635–651, 2000.

[8] T. Bächler, V. Gerdt, M. Lange-Hegermann, and D. Robertz. Algorithmic Thomas decomposition of algebraic and differential systems. *Journal of Symbolic Computation*, 47(10):1233–1266, 2012.

[9] G. Bard. *Algebraic cryptanalysis*. Springer, 2009.

[10] M. Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie*. PhD thesis, Université Pierre et Marie Curie, France, 2004.

[11] M. Bardet, J.-C. Faugère, and B. Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *International Conference on Polynomial System Solving - ICPSS*, pages 71 –75, 2004.

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

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

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

[15] L. Bettale, J.-C. Faugère, and L. Perret. Solving polynomial systems over finite fields: Improved analysis of the hybrid approach. In *Proceedings of ISSAC 2012*, pages 67–74. ACM Press, 2012.

[16] F. Boulier, F. Lemaire, and M. Moreno Maza. Well known theorems on triangular systems and the D5 principle. In *Proceedings of Transgressive Computing 2006*, pages 79–91, 2006.

[17] M. Bras-Amorós and M. O'Sullivan. The correction capability of the Berlekamp–Massey–Sakata algorithm with majority voting. *Applicable Algebra in Engineering, Communication and Computing*, 17(5):315–335, 2006.

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

[19] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal*. PhD thesis, Universitat Innsbruck, Austria, 1965.

[20] B. Buchberger. A criterion for detecting unnecessary reductions in the construction of Gröbner-bases. *Symbolic and Algebraic Computation*, pages 3–21, 1979.

[21] B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In N. Bose, editor, *Multidimensional Systems Theory*, pages 184–232. Springer, New York, 1985.

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

[23] F. Chai, X.-S. Gao, and C. Yuan. A characteristic set method for solving Boolean equations and applications in cryptanalysis of stream ciphers. *Journal of Systems Science and Complexity*, 21(2):191–208, 2008.

[24] X. Chen, I. Reed, T. Helleseth, and T. Truong. Use of Gröbner bases to decode binary cyclic codes up to the true minimum distance. *IEEE Transactions on Information Theory*, 40(5):1654–1661, 1994.

[25] S.-C. Chou. *Mechanical Geometry Theorem Proving*. Reidel, Dordrecht, 1988.

[26] S.-C. Chou and X.-S. Gao. Ritt-Wu's decomposition algorithm and geometry theorem proving. In M. E. Stickel, editor, *10th International Conference on Automated Deduction*, volume 449 of *LNAI*, pages 207–220. Springer, Berlin, 1990.

[27] S.-C. Chou and X.-S. Gao. Automated reasoning in differential geometry and mechanics using the characteristic set method. *Journal of Automated Reasoning*, 10(2):173–189, 1993.

[28] M. Cimpoeas. Generic initial ideal for complete intersections of embedding dimension three with strong Lefschetz property. *Bulletin Mathématique de la Société des Sciences Mathématiques de Roumanie. Nouvelle Série*, 50 (98)(1):33–66, 2007.

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

[30] G. Collins and A. Akritas. Polynomial real root isolation using Descarte's rule of signs. In *Proceedings of the third ACM symposium on symbolic and algebraic computation*, pages 272–275. ACM Press, 1976.

[31] P. Conti and C. Traverso. Buchberger algorithm and integer programming. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 130–139. Springer, 1991.

[32] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 392–407. Springer, Berlin, 2000.

[33] N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Y. Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 267–287. Springer, Berlin, 2002.

[34] N. T. Courtois and G. V. Bard. Algebraic cryptanalysis of the data encryption standard. In S. D. Galbraith, editor, *Cryptography and Coding*, pages 152–169. Springer, Berlin, 2007.

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

[36] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer Verlag, 1998.

[37] X. Dahan, M. Moreno Maza, E. Schost, W. Wu, and Y. Xie. Lifting techniques for triangular decompositions. In *Proceedings of ISSAC 2005*, pages 108–115. ACM Press, 2005.

[38] X. Dahan, M. Moreno Maza, É. Schost, and Y. Xie. On the complexity of the D5 principle. In *Transgressive Computing 2006*, pages 149–168, 2006.

[39] J. Davenport and B. Trager. Factorization over finitely generated fields. In *Proceedings of the fourth ACM symposium on symbolic and Algebraic computation*, pages 200–205. ACM Press, 1981.

[40] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, 1962.

[41] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3-1-4 — A computer algebra system for polynomial computations. 2012. http://www.singular.uni-kl.de.

[42] J. Della Dora, C. Dicrescenzo, and D. Duval. About a new method for computing in algebraic number fields. In *EUROCAL'85*, pages 289–290. Springer, 1985.

[43] S. Dellière. DM Wang simple systems and dynamic constructible closure. *Rapport de Recherche*, (2000-16), 2000.

[44] J. Ding, J. E. Gower, and D. S. Schmidt. *Multivariate Public Key Cryptosystems*. Springer, New York, 2006.

[45] D. Duval. Algebraic numbers: An example of dynamic evaluation. *Journal of Symbolic Computation*, 18(5):429–445, 1994.

[46] C. Eder and J. Perry. F5C: A variant of Faugère's F5 algorithm with reduced Gröbner bases. *Journal of Symbolic Computation*, 45(12):1442–1458, 2010.

[47] N. Eén and N. Sörensson. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing*, pages 333–336. Springer, 2004.

[48] P. Elias. List decoding for noisy channels. Technical Report 355, Research Laboratory of Electronics, Massachusetts Institute of Technology, 1957.

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

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

[51] J.-C. Faugère, P. Gaudry, L. Huot, and G. Renault. Fast change of ordering with exponent ω. *ACM Communnications in Computer Algebra*, 46:92–93, 2012.

[52] J.-C. Faugère, P. Gaudry, L. Huot, and G. Renault. Using symmetries in the index calculus for elliptic curves discrete logarithm. Cryptology ePrint Archive, Report 2012/199, 2012.

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

[54] J.-C. Faugère and A. Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using Gröbner bases. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *LNCS*, pages 44–60, Berlin, 2003. Springer.

[55] J.-C. Faugère and C. Mou. Fast algorithm for change of ordering of zero-dimensional Gröbner bases with sparse multiplication matrices. In *Proceedings of ISSAC 2011*, pages 115–122. ACM Press, 2011.

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

[57] J.-C. Faugère, M. Safey El Din, and P.-J. Spaenlehauer. Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1,1): Algorithms and complexity. *Journal of Symbolic Computation*, 46(4):406–437, 2011.

[58] J.-C. Faugère, M. Safey El Din, and P.-J. Spaenlehauer. Critical points and Gröbner bases: the unmixed case. In *Proceedings of ISSAC 2012*, pages 162–169. ACM Press, 2012.

[59] G. Feng and T. Rao. Decoding algebraic-geometric codes up to the designed minimum distance. *IEEE Transactions on Information Theory*, 39(1):37–45, 1993.

[60] P. Fitzpatrick. On the key equation. *IEEE Transactions on Information Theory*, 41(5):1290–1302, 1995.

[61] P. Fitzpatrick. Solving a multivariable congruence by change of term order. *Journal of Symbolic Computation*, 24:575–590, 1997.

[62] E. Fortuna, P. Gianni, and B. Trager. Derivations and radicals of polynomial ideals over fields of arbitrary characteristic. *Journal of Symbolic Computation*, 33(5):609–625, 2002.

[63] K. Fukuda, A. Jensen, and R. Thomas. Computing Gröbner fans. *Mathematics of Computation*, 76(260):2189–2212, 2007.

[64] X.-S. Gao and Z. Huang. Characteristic set algorithms for equation solving in finite fields. *Journal of Symbolic Computation*, 47(6):655–679, 2012.

[65] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, San Francisco, 1979.

[66] K. Geddes, S. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic, Boston, 1992.

[67] P. Gianni. Properties of Gröbner bases under specializations. In J. Davenport, editor, *Eurocal '87*, volume 378 of *LNCS*, pages 293–297. Springer, Berlin–Heidelberg, 1989.

[68] P. Gianni and B. Trager. Square-free algorithms in positive characteristic. *Applicable Algebra in Engineering, Communication and Computing*, 7(1):1–14, 1996.

[69] V. D. Goppa. Codes on algebraic curves. *Soviet Math. Dokl*, 24(1):170–172, 1981.

[70] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[71] C. Heegard, J. Little, and K. Saints. Systematic encoding via Gröbner bases for a class of algebraic-geometric Goppa codes. *IEEE Transactions on Information Theory*, 41(6):1752–1761, 1995.

[72] T. Høholdt, J. van Lint, and R. Pellikaan. *Algebraic Geometry Codes*. Elsevier, Amsterdam, 1998.

[73] Z. Huang and D. Lin. Attacking Bivium and Trivium with the characteristic set method. In A. Nitaj and D. Pointcheval, editors, *Progress in Cryptology–AFRICACRYPT 2011*, pages 77–91. Springer, 2011.

[74] E. Hubert. Notes on triangular sets and triangulation-decomposition algorithms I: Polynomial systems. In F. Winkler and U. Langer, editors, *Symbolic and Numerical Scientific Computation*, volume 2630 of *LNCS*, pages 143–158, Berlin, 2003. Springer.

[75] S. M. M. Javadi and M. B. Monagan. On factorization of multivariate polynomials over algebraic number and function fields. In *Proceedings of ISSAC 2009*, pages 199–206. ACM Press, 2009.

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

[77] M. Kalkbrener. A generalized Euclidean algorithm for computing triangular representations of algebraic varieties. *Journal of Symbolic Computation*, 15(2):143–167, 1993.

[78] M. Kalkbrener. Algorithmic properties of polynomial rings. *Journal of Symbolic Computation*, 26(5):525–581, 1998.

[79] E. Kaltofen. Polynomial factorization: A success story. In *Proceedings of ISSAC 2003*, pages 3–4. ACM Press, 2003.

[80] D. Kapur. Using Gröbner bases to reason about geometry problems. *Journal of Symbolic Computation*, 2(4):399–408, 1986.

[81] S. A. Kauffman. *The Origin of Order: Self-Organization and Selection in Evolution*. Oxford University Press, New York, 1993.

[82] C. Kelley. *Solving Nonlinear Equations with Newton's Method*, volume 1. Society for Industrial Mathematics, 1987.

[83] G. Kemper. The calculation of radical ideals in positive characteristic. *Journal of Symbolic Computation*, 34(3):229–238, 2002.

[84] R. Koetter and A. Vardy. Algebraic soft-decision decoding of reed-solomon codes. *IEEE Transactions on Information Theory*, 49(11):2809–2825, 2003.

[85] M. Kreuzer and L. Robbiano. *Computational Commutative Algebra 2*, volume 1. Springer Verlag, 2005.

[86] S. Lang. *Algebra*, volume 211 of *Graduate Texts in Mathematics*. Springer, New York, Revised 3rd edition, 2002.

[87] R. Laubenbacher and B. Stigler. A computational algebra approach to the reverse engineering of gene regulatory networks. *Journal of Theoretical Biology*, 229:523–537, 2004.

[88] R. Laubenbacher and B. Sturmfels. Computer algebra in systems biology. *American Mathmetical Monthly*, 116(10):882–891, 2009.

[89] D. Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In *Computer Algebra, EUROCAL' 83*, pages 146–156. Springer, 1983.

[90] D. Lazard. A new method for solving algebraic systems of positive dimension. *Discrete Applied Mathematics*, 33(1–3):147–160, 1991.

[91] D. Lazard. Solving zero-dimensional algebraic systems. *Journal of symbolic computation*, 13(2):117–131, 1992.

[92] K. Lee and M. O'Sullivan. List decoding of Reed–Solomon codes from a Gröbner basis perspective. *Journal of Symbolic Computation*, 43(9):645–658, 2008.

[93] F. Lemaire, M. Moreno Maza, and Y. Xie. The RegularChains library in Maple 10. In I. Kotsireas, editor, *Maple Conference 2005*, pages 355–368. Maplesoft, Waterloo, 2005.

[94] B. Li and D. Wang. An algorithm for transforming regular chain into normal chain. In D. Kapur, editor, *Computer Mathematics*, volume 5081 of *LNCS*, pages 236–245. Springer, Berlin – Heidelberg, 2008.

[95] X. Li, M. Moreno Maza, and W. Pan. Computations modulo regular chains. In *Proceedings of ISSAC 2009*, pages 239–246. ACM Press, 2009.

[96] X. Li, M. Moreno Maza, and É. Schost. Fast arithmetic for triangular sets: From theory to practice. In *Proceedings of ISSAC 2007*, pages 269–276. ACM Press, 2007.

[97] X. Li, C. Mou, W. Niu, and D. Wang. Stability analysis for discrete biological models using algebraic methods. *Mathematics in Computer Science*, 5:247–262, 2012.

[98] X. Li, C. Mou, and D. Wang. Decomposing polynomial sets into simple sets over finite fields: The zero-dimensional case. *Computers & Mathematics with Applications*, 60(11):2983–2997, 2010.

[99] X. Li and D. Wang. Simple decomposition of polynomial sets over finite fields (in Chinese). *Journal of Systems Science and Mathematical Sciences*, 32(1):15–26, 2012.

[100] Q. Liao. Equation solving in robotics and mechanisms. In X.-S. Gao and D. Wang, editors, *Mathematics Mechanization and Applications*, pages 433–460. Academic Press, London, 2000.

[101] R. Lidl and H. Niederreiter. *Finite Fields*. Addison-Wesley, Reading, Mass., 1983.

[102] Z. Lin, L. Xu, and N. Bose. A tutorial on Gröbner bases with applications in signals and systems. *IEEE Transactions on Circuits and Systems I*, 55(1):445–461, 2008.

[103] M. Liu, D. Lin, and D. Pei. Fast algebraic attacks and decomposition of symmetric boolean functions. *IEEE Transactions on Information Theory*, 57(7):4817–4821, 2011.

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

[105] F. Macwilliams and N. Sloane. *The Theory of Error-correcting Codes*. North-Holland, Amsterdam, l977.

[106] R. Matsumoto. Computing the radical of an ideal in positive characteristic. *Journal of Symbolic Computation*, 32(3):263–271, 2001.

[107] T. Matsumoto and H. Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In C. G. Günther, editor, *Advances in Cryptology — EUROCRYPT 88*, volume 330 of *LNCS*, pages 419–453. Springer, Berlin, 1988.

[108] E. Mayr and A. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in mathematics*, 46(3):305–329, 1982.

[109] B. Mishra. *Algorithmic Algebra*. Texts and Monographs in Computer Science. Springer, New York, 1993.

[110] T. Moh. A public key system with signature and master key functions. *Communications in Algebra*, 27(5):2207–2222, 1999.

[111] T. Mora. *Solving Polynomial Equation Systems II: Macaulay's Paradigm and Gröbner Technology*, volume 2. Cambridge University Press, 2005.

[112] M. Moreno Maza. On triangular decompositions of algebraic varieties. Technical Report 4/99, NAG, UK. Presented at the MEGA-2000 Conference, Bath, UK.

[113] M. Moreno Maza and R. Rioboo. Polynomial GCD computations over towers of algebraic extensions. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 365–382. Springer, 1995.

[114] G. Moreno-Socías. *Autour de la fonction de Hilbert-Samuel (escaliers d'idéaux polynomiaux)*. PhD thesis, Ecole Polytechnique, France, 1991.

[115] G. Moreno-Socías. Degrevlex Gröbner bases of generic complete intersections. *Journal of Pure and Applied Algebra*, 180(3):263–283, 2003.

[116] A. Morgan. *Solving Polynominal Systems Using Continuation for Engineering and Scientific Problems*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[117] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001.

[118] C. Mou, D. Wang, and X. Li. Decomposing polynomial sets into simple sets over finite fields: The positive-dimensional case. *Theoretical Computer Science*, (468):102–113, 2013.

[119] R. Nielsen and T. Hholdt. Decoding Reed-Solomon codes beyond half the minimum distance. In H. S. Johannes Buchmann, Tom Hoeholdt and H. Tapia-Recillas, editors, *Coding Theory, Cryptography and Related Areas*, pages 221–236. Springer-Verlag, 2000.

[120] W. Niu and D. Wang. Algebraic approaches to stability analysis of biological systems. *Mathematics in Computer Science*, 1(3):507–539, 2008.

[121] M. Noro. Modular dynamic evaluation. In *Proceedings of ISSAC 2006*, pages 262–268. ACM Press, 2006.

[122] M. Noro and T. Takeshima. Risa/Asir–a computer algebra system. In *Proceedings of ISSAC 1992*, pages 387–396. ACM Press, 1992.

[123] M. Noro and K. Yokoyama. Factoring polynomials over algebraic extension fields. *Josai Information Science Researches*, 9:11–33, 1997.

[124] M. Noro and K. Yokoyama. Implementation of prime decomposition of polynomial ideals over small finite fields. *Journal of Symbolic Computation*, 38(4):1227–1246, 2004.

[125] H. O'Keeffe and P. Fitzpatrick. Gröbner basis solutions of constrained interpolation problems. *Linear Algebra and its Applications*, 351–352:533–551, 2002.

[126] L. Pachter and B. Sturmfels. *Algebraic Statistics for Computational Biology*. Cambridge University Press, Cambridge, 2005.

[127] V. Pan. Solving a polynomial equation: Some history and recent progress. *SIAM review*, pages 187–220, 1997.

[128] K. Pardue. Generic sequences of polynomials. *Journal of Algebra*, 324(4):579–590, 2010.

[129] H. A. Park and G. Regensburger. *Gröbner bases in Control Theory and Signal Processing*, volume 3 of *Radon Series on Computational and Applied Mathematics*. De Gruyter, 2007.

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

[131] J. Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. In D. Coppersmith, editor, *Advances in Cryptology — CRYPTO 95*, volume 963 of *LNCS*, pages 248–261. Springer, Berlin, 1995.

[132] J. Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT 96*, volume 1070 of *LNCS*, pages 33–48. Springer, Berlin, 1996.

[133] W. Plesken. Counting solutions of polynomial systems via iterated fibrations. *Archiv der Mathematik*, 92(1):44–56, 2009.

[134] J. F. Ritt. *Differential Algebra*. American Mathematical Society, New York, 1950.

[135] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.

[136] F. Rouillier and P. Zimmermann. Efficient isolation of polynomial's real roots. *Journal of Computational and Applied Mathematics*, 162(1):33–50, 2004.

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

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

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

[140] S. Sakata. Gröbner bases and coding theory. In B. Buchberger and F. Winkler, editors, *Gröbner Bases and Applications*, volume 251 of *London Mathematical Society Lecture Note Series*, pages 205–220. Birkhauser, Cambridge University Press, 1998.

[141] A. Seidenberg. Constructions in algebra. *Transactions of the American Mathematical Society*, 197:273–313, 1974.

[142] M. Soos, K. Nohl, and C. Castelluccia. Extending SAT solvers to cryptographic problems. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, pages 244–257. Springer-Verlag, 2009.

[143] P.-J. Spaenlehauer. *Résolution de systèm multi-homogènes et déterminantiels: Algorithms, complexities and applications*. PhD thesis, Université Pierre et Marie Curie, France, 2012.

[144] A. Steel. Conquering inseparability: Primary decomposition and multivariate factorization over algebraic function fields of positive characteristic. *Journal of Symbolic Computation*, 40(3):1053–1075, 2005.

[145] B. Stigler and A. Veliz-Cuba. Network topology as a driver of bistability in the *lac* operon. *Arxiv preprint arXiv:0807.3995*, 2008.

[146] Y. Sun and D. Wang. An efficient algorithm for factoring polynomials over algebraic extension field. *Science in China, Series A: Mathematics*. Accepted.

[147] Y. Sun and D. Wang. A generalized criterion for signature related Gröbner basis algorithms. In *Proceedings of ISSAC 2011*, pages 337–344. ACM Press, 2011.

[148] T. Tamura and T. Akutsu. Algorithms for singleton attractor detection in planar and nonplanar AND/OR Boolean networks. *Mathematics in Computer Science*, 2(3):401–420, 2009.

[149] J. M. Thomas. *Differential Systems*. American Mathematical Society, New York, 1937.

[150] R. R. Thomas. A geometric Buchberger algorithm for integer programming. *Mathematics of Operations Research*, 20(4):864–884, 1995.

[151] P. Trifonov. Efficient interpolation in the guruswamisudan algorithm. *IEEE Transactions on Information Theory*, 56(9):4341–4349, 2010.

[152] J. Verschelde. Homotopy Methods for Solving Polynomial Systems, tutorial at ISSAC 2005, July 24–27, 2005. Available at http://www.math.uic.edu/~jan/tutorial.pdf.

[153] J. Von Zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2003.

[154] D. Wang. On Wu's method for proving constructive geometric theorems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 419–424. Morgan Kaufmann, San Fransisco, 1989.

[155] D. Wang. An elimination method for polynomial systems. *Journal of Symbolic Computation*, 16(2):83–114, 1993.

[156] D. Wang. An implementation of the characteristic set method in Maple. *Automated practical reasoning: algebraic approaches*, pages 187–201, 1995.

[157] D. Wang. Decomposing polynomial systems into simple systems. *Journal of Symbolic Computation*, 25(3):295–314, 1998.

[158] D. Wang. Computing triangular systems and regular systems. *Journal of Symbolic Computation*, 30(2):221–236, 2000.

[159] D. Wang. *Elimination Methods*. Springer Verlag, 2001.

[160] D. Wang and D. Lin. A method for multivariate polynomial factorization over successive algebraic extension fields. In W. Lin, D. Li and Y. Yu, editors, *Mathematics and Mathematics-Mechanization*, pages 138–172, Jinan, 2001. Shandong Education Publishing House.

[161] D. Wang and B. Xia. Stability analysis of biological systems with real solution classification. In *Proceedings of ISSAC 2005*, pages 354–361. ACM Press, 2005.

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

[163] F. Winkler. *Polynomial Algorithms in Computer Algebra*. Texts and Monographs in Symbolic Computation. Springer, Berlin, 1996.

[164] W.-T. Wu. Basic principles of mechanical theorem proving in elementary geometries. *Journal of Automated Reasoning*, 2(3):221–252, 1986.

[165] W.-T. Wu. On zeros of algebraic equations: An application of Ritt principle. *Kexue Tongbao*, 31(1):1–5, 1986.

[166] W.-T. Wu and X.-S. Gao. Mathematics mechanization and applications after thirty years. *Frontiers in Computer Science*, 1(1):1–8, 2007.

[167] L. Yang and J.-Z. Zhang. Searching dependency between algebraic equations: An algorithm applied to automated reasoning. In J. Johnson, S. McKee, and A. Vella, editors, *Artificial Intelligence in Mathematics*, pages 147–156, Oxford, 1994. Oxford University Press.

Curriculum Vitae

Student Dormitory 9–215, Beihang University
XueYuan Road No.37, HaiDian District
Beijing 100191, China

Chenqi.Mou@gmail.com
(+86) 138 1142 6823
http://www-polsys.lip6.fr/~mou/

EDUCATION

Ph.D. in *Applied Mathematics / Computer Science* May 2013
Beihang University, Beijing, China / Université Pierre et Marie Curie, Paris, France

- Supervisors: Dongming Wang / Jean-Charles Faugère
- Subject: Solving Polynomial Systems over Finite Fields: Algorithms, Implementation and Applications
- Research Field: Symbolic Computation

Bachelor of Science in *Mathematics and Applied Mathematics* Jul. 2007
Beihang University, Beijing, China

- Top 5 out of 120

PUBLICATIONS

Book

[1] D. Wang, **C. Mou**, X. Li, J. Yang, M. Jin, and Y. Huang. *Polynomial Algebra* (in Chinese), Higher Education Press, Beijing, 2011

Journal Papers

[2] **C. Mou**, D. Wang, and X. Li. Decomposing polynomial sets into simple sets over finite fields: The positive-dimensional case. *Theoretical Computer Science*, 2013, 468: 102–113

[3] **C. Mou**. Design of termination criterion of BMS algorithm for lexicographical ordering (in Chinese). *Journal of Computer Applications*, 2012, 32(11): 2977–2980

[4] X. Li, **C. Mou**, W. Niu, and D. Wang. Stability analysis for discrete biological models using algebraic methods. *Mathematics in Computer Science*, 2011, 5: 247–262

[5] X. Li, **C. Mou**, and D. Wang. Decomposing polynomial sets into simple sets over finite fields: The zero-dimensional case. *Computers and Mathematics with Applications*, 2010, 60: 2983–2997

[6] J.-C. Faugère and **C. Mou**. Sparse FGLM algorithms. Preprint arXiv:1304.1238, 2013

[7] **C. Mou** and W. Niu. Application of triangular set methods to detection of steady states and their numbers for finite biological models (in Chinese). *Computer Applications and Software*, accpeted

[8] **C. Mou** and Dongming Wang. Squarefree decomposition and factorization over unmixed products of field extensions. In preparation

Conference Papers

[9] J.-C. Faugère and **C. Mou**. Fast algorithm for change of ordering of zero-dimensional Gröbner bases with sparse multiplication matrices. *International Symposium on Symbolic and Algebraic Computation 2011* (*ISSAC 2011*). San Jose, USA, June 8–11

- Top conference in Theoretical Computer Science

[10] X. Li, **C. Mou**, W. Niu, and D. Wang. Stability analysis for discrete biological models using algebraic methods. *Mathematical Aspects of Computer and Information Sciences 2009* (*MACIS 2009*). Fukuoka, Japan, December 14–17

ACADEMIC ACTIVITIES

Contributed talks:

- Simple Triangular Sets: from Q to F_q
  Seminar on Symbolic Computations, Beijing, China May 2012
- Fast Algorithm for Change of Ordering of Zero-dimensional Gröbner Bases with Sparse Multiplication Matrices
  International Workshop on Certified and Reliable Computation, NanNing, China Jul. 2011
- Fast Algorithm for Change of Ordering of Zero-dimensional Gröbner Bases with Sparse Multiplication Matrices
  International Symposium on Symbolic and Algebraic Computation, San Jose, USA Jun. 2011

Member of Local Arrangements for

- Fourth International Conference on Mathematical Aspects of Computer and Information Sciences
  Beihang University, Beijing, China Oct. 2011
- First International Conference on Symbolic Computation and Cryptography
  Beihang University, Beijing, China Apr. 2008

External reviewer for

- SCIENCE CHINA Information Sciences
- Mathematics in Computer Science
- First International Conference on Symbolic Computation and Cryptography

Participant in

- Summer School on Symbolic Computation 2009
  University of Electronic Science and Technology, Chengdu, China Aug. 2009
- International Conference on Mathematics Mechanization
  Chinese Academy of Science, Beijing, China May 2009

- NUS–PKU Joint Summer Programme on Mathematical Modeling
  National University of Singapore, Singapore / Peking University, Beijing, China Jun.–Jul. 2006

SELECTED HONORS

- Scholarship from Chinese Scholarship Council to support the study in France 2009–2011
- Second Prize of China Graduate Mathematical Contest in Modeling 2008
- Meritorious Winner of Mathematical Contest in Modeling (COMAP in USA) 2007
- Second Prize of China Undergraduate Mathematical Contest in Modeling 2006
- First Prize of National English Contest for College Students in China 2005

LANGUAGES

Chinese: **Native**

English: **Fluent**

- Translation Proficiency Qualification Certificate of the People's Republic of China level III

COMPUTER SKILLS

Magma, Maple, Matlab, C++, Linux, LaTeX, Emacs