Eindhoven University of Technology
Department of Mathematics and Computer Science

Coding Theory: A Gröbner Basis Approach

Master's Thesis by
D.W.C. Kuijsters

Supervised by
Dr. G.R. Pellikaan

February 6, 2017

Abstract

One of the central topics in algorithmic coding theory is the design of efficient decoding algorithms. For a large subclass of cyclic codes, the so-called BCH codes, such efficient algorithms are known. The number of errors which a code is capable of correcting is entirely specified by its minimum distance. If we write d for the minimum distance, then a code should be able to correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors. The problem with the algorithms used for decoding BCH codes is that they only correct up to $\lfloor \frac{d_{BCH}-1}{2} \rfloor$ errors where d_{BCH} is the BCH bound. Often the real minimum distance is much larger than this lower bound. It would be satisfactory to come up with algorithms for correcting up to the true minimum distance.
It is possible to translate the problem of decoding into a parametrized system of equations. Solving this system is then equivalent to solving the decoding problem. Related to this is the problem of computing the (often unknown) minimum distance. It turns out that the same system of equations can be used to solve this problem.
The polynomials constituting a system of equations determine an ideal. Every ideal has a finite set of "small" generators called a Gröbner basis. From this basis the solution set to the original system can easily be read off. In his 1965 dissertation Buchberger presented the first algorithm for computing such a basis. Unfortunately, it turns out that this is a computationally hard problem. Over the years, many variations improving on Buchberger's algorithm have been presented, including Faugère's F5 algorithm and, more recently, Gao, Volny, and Wang's GVW algorithm. In general, useless reductions, i.e., reductions to zero, are the primary bottleneck in Gröbner basis computations. Therefore, both algorithms incorporate a number of criteria based on the notion of a signature to detect these useless reductions in advance. If the polynomials constituting a system form a semi-regular sequence, then no useless reductions are performed at all. Unfortunately, not all systems are given by a semi-regular sequence, hence much attention has been devoted to exploiting the algebraic structure of the system under consideration in order to speed up the algorithms.

Preface

This thesis is the result of my graduation project in completion of the Master of Science degree. The graduation project was conducted under the supervision and guidance of Ruud Pellikaan within the Coding Theory and Cryptology group at Eindhoven University of Technology.

The thesis consists of two parts. The first part deals with Gröbner basis techniques. In particular, we will discuss the abstract framework and two particular algorithms in great detail. The second part deals with coding theory. In particular, we will discuss how the problem of decoding and finding the minimum distance can be translated into a system of polynomial equations. In turn, the techniques discussed in the first part of this thesis can be used to solve these problems.

- Chapter 1 gives a very brief overview of the background expected from the reader.
- Chapter 2 introduces the notion of a Gröbner basis. Moreover, we give a brief introduction to the language of algebraic geometry. With it, we formalize what it means to solve a system and give a number of invariants of the system, such as its dimension and degree. Finally, we present a number of algebraic tools which are useful for analyzing these invariants.
- Chapter 3 extends the notion of a Gröbner basis by incorporating signatures. These are pieces of data associated with a polynomial encoding how that polynomial depends on a given basis.
- Chapter 4 introduces the first example of a signature-based Gröbner basis algorithm, the Matrix-F5 algorithm. Using the F5-criterion, it is able to predict useless reductions to zero. Moreover, it shows how one can use fast linear algebra algorithms to speed up the reduction process. In addition to describing the original Matrix-F5 algoritm, we discuss a number of extensions that appear in the literature. Finally, we show for the first time how the so-called syzygy criterion can be incorporated into the algorithm.
- Chapter 5 presents a relatively new signature-based Gröbner basis algorithm, the GVW algorithm. The novel idea in this algorithm is to look at a larger module than just the ideal itself. Instead of discarding a zero reduction, it stores the signature corresponding to the reduction and uses it to predict future zero reductions.
- Chapter 6 discusses a small number of experiments related to the algorithms that we have implemented.

- Chapter 7 gives an introduction to linear codes, the second part of this thesis. Moreover, it discusses the problem of decoding.
- Chapter 8 gives an overview of cyclic codes, which are linear codes with more structure, and presents a number of ways of translating the decoding problem for cyclic codes to a system of polynomial equations.
- Chapter 9 extends the ideas of the previous chapter to a method of translating the decoding problem for general linear codes into a system of quadratic equations. Finally, we will look at two possible applications.
- Chapter 10 describes the link between linear codes and a particular type of ideal which appears quite often in applied mathematics, a toric ideal. At the end of the chapter, we present a heuristic for the decoding problem. Unfortunately, this heuristic is not very practical.
- Chapter 11 discusses a number of experiments related to the various ways of translating the decoding problem into a system of equations.
- The appendix contains the implementation in Magma of several algorithms appearing throughout this thesis.

I would like to express my gratitude to Wieb Bosma, Cees Jansen, and Hans Sterk for taking part in my assessment committee. A special word of thanks goes to my graduation supervisor Ruud Pellikaan for the many useful comments that I have received over the span of this project.

Rietheven, the Netherlands. Daniël Kuijsters
February 6, 2017

Contents

Chapter 1

Preliminaries

This chapter will serve to fix the conventions and notation we will be using. Let $\mathbb{N}$ denote the set of non-negative integers.

We are assuming that the reader has taken undergraduate courses in linear and abstract algebra. In particular, the reader is assumed to be familiar with basic algebraic structures such as groups, rings, fields, vector spaces, and their properties.

A notion which is generally not taught at the undergraduate level is that of a module M over a ring A. Its definition is identical to that of a vector space over a field. In other words, the notion of a module over a ring generalizes that of a vector space over a field. We will say that M is an A-module. In particular, an A-module homomorphism is the same as an A-linear map. An A-module is finitely generated if A is equal to the set of all A-linear combinations of a set $\{m_1, \ldots, m_k\}$ of finite cardinality. A free A-module is one that is isomorphic to $\oplus_{i \in I} M_i$ where each $M_i \simeq A$. It follows that a finitely generated free A-module is isomorphic to A^k.

A general field will be denoted by k. An important class of examples is the class of finite fields, i.e., fields with a finite number of elements. We will write $\mathbb{F}_q$ for the finite field of order q. Here $q = p^l$ for a prime number $p \in \mathbb{N}$ and $l > 0$. We will write $\mathbb{F}_p$ for its prime field.

The characteristic of a field k is the smallest $n \in \mathbb{N}$ such that $n \cdot 1 = 0$ in k. The characteristic of k is either 0 or a prime p. So the characteristic of $\mathbb{F}_q$ is p if $q = p^l$. Recall that all finite field extensions of $\mathbb{F}_q$ are of the form $\mathbb{F}_{q^m}$ for some $m \in \mathbb{N}$.

A field k is called algebraically closed if every non-constant polynomial in $k[x]$ has a root contained in k. Every field k is contained in a field $\overline{k}$ that is algebraically closed. We call $\overline{k}$ the algebraic closure of k.

To us, the most important example of a ring is the polynomial ring over k in the indeterminates $x_1, \ldots, x_n$, denoted $k[x_1, \ldots, x_n]$. From now on, we will write A_n for $k[x_1, \ldots, x_n]$. This ring is commutative, i.e., $fg = gf$ for all $f, g \in A_n$. Moreover, it is an integral domain, i.e., if $f, g \in A_n$ are both nonzero, then their product is nonzero.

The elements of A_n are called polynomials. To define them, we start by first defining mono-

mials. A monomial in $x_1, \ldots, x_n$ is a product of the form

$$x_1^{a_1} \cdot \cdots \cdot x_n^{a_n}$$

where we require that each a_i is a non-negative integer.

Given a monomial $x_1^{a_1} \cdot \cdots \cdot x_n^{a_n}$, we call $\alpha = (a_1, \ldots, a_n)$ its exponent vector. Hence, for simplicity of notation, we often write x^α for $x_1^{a_1} \cdot \cdots \cdot x_n^{a_n}$. The (total) degree of a monomial $x^\alpha = x_1^{a_1} \cdot \cdots \cdot x_n^{a_n}$ is equal to the sum of the exponents $|\alpha| := a_1 + \cdots + a_n$. Throughout this text, we will let T_n denote the monoid of all monomials in $x_1, \ldots, x_n$, i.e., $T_n = \{x^\alpha : \alpha \in \mathbb{N}^n\}$.

A polynomial f in $x_1, \ldots, x_n$ is a finite k-linear combination of monomials, i.e., it is of the form

$$f = \sum_\alpha c_\alpha x^\alpha, c_\alpha \in k$$

where the sum is over a finite number of n-tuples α of non-negative integers. Next, we need the notion of the formal derivative of a polynomial. Let A be a ring. If $a(x) = \sum_{i=0}^n a_i x^i$ is a polynomial in $A[x]$, then the formal derivative of $a(x)$ is defined as $a'(x) = \sum_{i=1}^n i a_i x^{i-1}$.

Given polynomials $f_1, \ldots, f_m$ in A_n they define a set of equations. Obviously, taking sums and multiplying the equations by arbitrary nonzero polynomials does not change the set of solutions. This inspires the following definition. Let $S \subseteq A_n$ and let $\langle S \rangle$ denote the collection

$$\langle S \rangle = \{p_1 f_1 + \cdots + p_m f_m : m \in \mathbb{N}, f_i \in S, p_i \in k[x_1, \ldots, x_n], \text{ for } i = 1, \ldots, m\}$$

If $S = \{f_1, \ldots, f_m\}$ we will also write $\langle f_1, \ldots, f_m \rangle$ for $\langle S \rangle$. Recall that the collection thus defined is an example of an ideal in A_n, i.e., a subgroup of A_n that is invariant under multiplication by polynomials.

An ideal I in A is called prime if A/I is an integral domain. An ideal I in A is called maximal if A/I is a field.

A syzygy on $(f_1, \ldots, f_m)$ is an element $(a_1, \ldots, a_m) \in A_n^m$ such that $\sum_{i=1}^m a_i f_i = 0$. In other words, it is a relation between the f_i.

Hilbert's basis theorem says that A_n is Noetherian. A ring A is called is Noetherian if every ideal in A is finitely generated. Equivalently, every ascending chain of ideals in A

$$I_1 \subseteq I_2 \subseteq \cdots$$

eventually stabilizes, i.e., there exists an $n \in \mathbb{N}$ such that $I_n = I_{n+1} = \cdots$.

Part I

Gröbner basis theory

Chapter 2

Classical Gröbner basis theory

The ideal membership problem can be formulated as follows: given $f, f_1, \ldots, f_m \in A_n$, is $f \in \langle f_1, \ldots, f_m \rangle$? Gröbner bases were introduced to solve exactly this problem. In some sense, a Gröbner basis is a set of "small" generators for the ideal. Unfortunately, the membership problem is EXPSPACE-Hard [Sud], hence we cannot hope to find a polynomial-time algorithm for computing a Gröbner basis.

2.1 Order theory

We would like to get a unique representation of the elements of A_n. To this end, we equip A_n with a so-called monomial order.

Definition 2.1.1 (Monomial order)**.** *A monomial order on A_n is a relation, denoted $>$, on the set $\mathbb{N}^n$, or equivalently on the set of monomials $x^\alpha, \alpha \in \mathbb{N}^n$ (via the map $\alpha \mapsto x^\alpha$), satisfying*

- *$>$ is a total order.*
- *$>$ is compatible with multiplication, i.e., if $\alpha > \beta$ and $\gamma \in \mathbb{N}^n$, then $\alpha + \gamma > \beta + \gamma$.*
- *$>$ is a well-order, i.e., every non-empty subset of $\mathbb{N}^n$ has a smallest element relative to $>$.*

We use the convention that $x^\alpha > 0$ for all $\alpha \in \mathbb{N}^n$. The following orders are the classic examples of monomial orders:

Definition 2.1.2 (Lexicographic order)**.** *$\alpha >_{lex} \beta$ if the leftmost nonzero entry of $\alpha - \beta$ is positive. We will write $x^\alpha >_{lex} x^\beta$ if $\alpha >_{lex} \beta$.*

Definition 2.1.3 (Degree reverse lexicographic order)**.** *$\alpha >_{degrevlex} \beta$ if either*

- *$|\alpha| > |\beta|$ or*
- *$|\alpha| = |\beta|$ and the rightmost nonzero entry of $\alpha - \beta$ is negative. We will write $x^\alpha >_{degrevlex} x^\beta$ if $\alpha >_{degrevlex} \beta$.*

The graded reverse lexicographic order is an example of what we call a degree compatible ordering, i.e., an ordering such that

$$|\alpha| > |\beta| \Rightarrow \alpha > \beta.$$

Next, we describe a monomial order that will come up when we talk about elimination.

Definition 2.1.4 (Elimination order)**.** *Suppose the variables in A_n are divided into two blocks, say $y = (y_1, \ldots, y_h)$ and $z = (z_1, \ldots, z_k)$, and let $>_1$ and $>_2$ be monomial orders on the first and second block respectively. We say that $>$ is an elimination order with respect to (y, z) if either*

- *$y^\alpha >_1 y^\gamma$ or*
- *$y^\alpha = y^\gamma$ and $z^\beta >_2 z^\delta$*

Thus, in an elimination order we compare variables in the first block and only look at the variables in the second block in the case of ties. An example of an elimination order is the lexicographic order.
Now that we have defined a monomial order, we can agree on some terminology that will come up over and over again.

Definition 2.1.5. *Let $f = \sum_\alpha c_\alpha x^\alpha$ be a nonzero polynomial in A_n and let $>$ be a monomial order.*

- *We call c_α the coefficient of the monomial x^α.*
- *If $c_\alpha \neq 0$, we call $c_\alpha x^\alpha$ a term of f.*
- *We call $\deg(f) = \max\{\alpha_1 + \cdots + \alpha_n : c_\alpha \neq 0\}$ the (total) degree of f.*
- *We call $\text{multideg}(f) = \max_>\{\alpha \in \mathbb{Z}^n_{\geq 0} : c_\alpha \neq 0\}$ the multidegree of f.*
- *The leading coefficient of f is $\text{lc}_>(f) = \text{lc}(f) = c_{\text{multideg}(f)}$.*
- *The leading monomial of f is $\text{lm}_>(f) = \text{lm}(f) = x^{\text{multideg}(f)}$.*
- *The leading term of f is $\text{lt}_>(f) = \text{lt}(f) = \text{lc}(f)\,\text{lm}(t)$.*
- *A polynomial with leading coefficient equal to 1 is called monic.*

We will define $\deg(0) = -1$ as we will need it later on.

2.2 Multivariate division

Let us go back to the ideal membership problem described at the beginning of this chapter. Suppose $f, f_1, \ldots, f_m \in k[x]$ were univariate polynomials. Then it is easy to solve the problem. Since $k[x]$ is a principal ideal domain the ideal $\langle f_1, \ldots, f_m \rangle$ is generated by $g = \gcd(f_1, \ldots, f_m)$. It follows that we only need to check whether f is a multiple of g which can be done by division with remainder. It is the concept of division that we want to generalize to A_n. First we will recall how univariate division works: suppose we wish to divide $f = a_0 + a_1 x + \cdots + a_n x^n$ by $g = b_0 + b_1 x + \cdots + b_m x^m$. Notice that the terms are ordered by

degree. We divide the leading term (with respect to this ordering) of f by the leading term of g, i.e., we calculate $q_i := \frac{a_n x^n}{b_m x^m}$ and recursively apply this principle to $f_i := f - q_i g$ and g. When the leading term of g no longer divides f_i we end the recursion and store $\sum_i q_i$ in q, which we call the quotient, and f_i in r, which we call the remainder. We will try to mimick this procedure in A_n:

Definition 2.2.1 (Top-reduction)**.** *Let f and g be polynomials in A_n. We say that f is top-reducible by g if $\mathrm{lm}(g)$ divides $\mathrm{lm}(f)$. The corresponding top-reduction is given by*

$$f - \frac{\mathrm{lt}(f)}{\mathrm{lt}(g)} g$$

The effect of a top-reduction is that the leading term of f is cancelled.

Let $F = \{f_1, \ldots, f_m\}$ be a set of polynomials in A_n. We say that f is top-reducible by F if there exists an $1 \leq i \leq m$ such that f is top-reducible by f_i. When no f_i top-reduces f we say that f is top-irreducible.
When f is top-irreducible, we may proceed to reduce $f - \mathrm{lt}(f)$. If there is a term of f that is divisible by a leading monomial of some f_i we say that f is reducible by F. When f is no longer reducible, we end up with a remainder r. However, this remainder is, in general, not unique, as it depends on the order of reductions.

Proposition 2.2.2. *Let $F = \{f_1, \ldots, f_m\}$ be a sequence of polynomials in A_n and fix a monomial order $>$. Every polynomial $f \in A_n$ can be written as a sum*

$$f = \sum_{i=1}^{m} q_i f_i + r$$

where $q_i, r \in A_n$ and either $q_i f_i = 0$ or $\mathrm{lt}(f) > \mathrm{lt}(q_i f_i)$ for $1 \leq i \leq m$. Moreover, we either have $r = 0$ or r is a linear combination of monomials that are not divisible by any $\mathrm{lt}(f_i)$ for $1 \leq i \leq m$. We will call r a remainder of f on division by F.

An algorithm computing the quotients and the remainder is given below.

input : $F = (f_1, \ldots, f_m)$ a sequence of polynomials in A_n, a polynomial $f \in A_n$, and a monomial order $>$
output: Polynomials $r, q_1, \ldots, q_m \in A_n$ such that $f = \sum_{i=1}^{m} q_i f_i + r$
begin
  $r := 0$;
  $h := f$;
  for $i = 1$ **to** m **do**
    $q_i := 0$;
  end
  while $h \neq 0$ **do**
    $j := 1$;
    $divided := false$;
    while $j \leq m \wedge \neg divided$ **do**
      if $\mathrm{lt}(f_j)$ *divides* $\mathrm{lt}(h)$ **then**
        $h := h - \frac{\mathrm{lt}(h)}{\mathrm{lt}(f_j)} \mathrm{lt}(f_j)$;
        $q_j := q_j + \frac{\mathrm{lt}(h)}{\mathrm{lt}(f_j)}$;
        $divided := true$
      else
        $j := j + 1$;
      end
    end
    if $\neg divided$ **then**
      $h := h - \mathrm{lt}(h)$;
      $r := r + \mathrm{lt}(h)$;
    end
  end
  return $r, q_1, \ldots, q_m$
end

Algorithm 1: Multivariate division

2.3 The notion of a Gröbner basis

Finally, we introduce the notion of a Gröbner basis for an ideal. This is a nice set of generators such that reduction by this set always leads to a unique remainder. First, we need a definition.

Definition 2.3.1. *Let $\{0\} \neq I$ be an ideal in A_n. The set of leading terms is defined as*

$$\mathrm{lm}(I) = \{\mathrm{lm}(f) : 0 \neq f \in I\}$$

The leading term ideal is the ideal $\langle \mathrm{lm}(I) \rangle$ generated by the set $\mathrm{lm}(I)$.

Definition 2.3.2 (Gröbner basis)**.** *A finite subset $G = \{g_1, \ldots, g_s\} \subseteq I$ is said to be a Gröbner basis for I (with respect to $>$) if it satisfies*

$$\langle \mathrm{lm}(g_1), \ldots, \mathrm{lm}(g_m) \rangle = \langle \mathrm{lm}(I) \rangle$$

Thus every nonzero polynomial in I is top-reducible by G.

Theorem 2.3.3. *Fix a monomial order $>$. Every ideal I has a Gröbner basis with respect to $>$. Moreover, if G is a Gröbner basis for the ideal I, then $I = \langle G \rangle$.*

Proof. See [CLO15] chapter 2, §5, corollary 6. □

A problem with the multivariate division algorithm is that, in general, the outcome depends on choices made during the algorithm. If the set of input polynomials forms a Gröbner basis for the ideal spannend by it then the outcome is, in fact, unique. This is the content of the following proposition.

Proposition 2.3.4. *Let $G = \{g_1, \ldots, g_s\}$ be a Gröbner basis for I and let $0 \neq f \in I$. Then there exists a unique $r \in A_n$ satisfying*

- *No term of r is divisble by any of $\mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_s)$, and*
- *There is a $g \in I$ such that $f = g + r$.*

This r is often called the normal form of f with respect to G and we denote it by $f \operatorname{rem} G$.

Proof. See [CLO15] chapter 2, §6, proposition 1. □

In general, we will write $f \operatorname{rem} F$ for the unique remainder on division by the ordered $m-$tuple $F = (f_1, \ldots, f_m)$. In the case that F is a Gröbner basis we have just seen that the order is irrelevant. We now have all the tools necessary to solve the membership problem. Let $f, f_1, \ldots, f_m \in A_n$ and recall that we want to know if $f \in \langle f_1, \ldots, f_m \rangle$. Compute a Gröbner basis G for $\langle f_1, \ldots, f_m \rangle$ and divide f by G. The remainder, which is unique, will answer the question.

Corollary 2.3.5. *Let $G = \{g_1, \ldots, g_s\}$ be a Gröbner basis for an ideal $I \subset A_n$ and $0 \neq f \in A_n$, then $f \in I$ if and only if $f \operatorname{rem} G = 0$.*

Proof. Suppose that $f \in I$. By proposition 2.3.4 $f \operatorname{rem} G = f - g$ for some $g \in I$. It follows that $f \operatorname{rem} G \in I$. However, no term of $f \operatorname{rem} G$ is divisble by any of $\mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_s)$. The fact that G is a Gröbner basis then implies that $f \operatorname{rem} G = 0$. To prove the converse, we assume that $f \operatorname{rem} G = 0$. Again, by proposition 2.3.4, we deduce that there exists a $g \in I$ such that $f = g + (f \operatorname{rem} G)$. Our assumption then implies that $f = g \in I$. □

Besides testing for membership, a Gröbner basis can also be used to do computations on the residue classes in A_n/I. It turns out that each residue class is a k-linear combination of the image in A_n/I of so-called standard monomials.

Definition 2.3.6 (Standard monomial)**.** *The monomials in the set* $\{x^\alpha : x^\alpha \notin \langle \mathrm{lm}(I)\rangle\}$ *are called standard.*

Theorem 2.3.7 (Macaulay's basis theorem)**.** *The residue classes of the standard monomials form a* k*-vector space basis for the quotient ring* A_n/I*.*

Proof. Let $f + I \in A_n/I$ and let $G = \{g_1, \ldots, g_s\}$ be a Gröbner basis for I. Take the element $(f \operatorname{rem} G) + I$ as a representative for $f + I$. By proposition 2.3.4 no term of $f \operatorname{rem} G$ is in $\langle \mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_s)\rangle$. Equivalently, $f \operatorname{rem} G$ is a k-linear combination of standard monomials. □

A Gröbner basis is, in general, not unique. We would like a unique representation of the ideal, relative to a fixed order. To this end, we give the following definition.

Definition 2.3.8 (Reduced Gröbner basis)**.** *A reduced Gröbner basis for an ideal* I *is a Gröbner basis* G *for* I *such that*

- $\mathrm{lc}(g) = 1$ *for all* $g \in G$*.*
- $g \operatorname{rem}(G \setminus \{g\}) = g$ *for all* $g \in G$*.*

It turns out that a reduced Gröbner basis is this unique representation.

Theorem 2.3.9. *Every ideal* $I \subseteq A_n$ *has a unique reduced Gröbner basis* G*.*

Proof. See [CLO15] chapter 2, §7, theorem 5. □

2.4 Buchberger's algorithm

Given the usefulness of Gröbner bases, we ask ourselves the question how to find such a basis. Buchberger, in his 1965 text, gave the first algorithm to compute a Gröbner basis. His algorithm relies heavily on the concept of the S-polynomial of two polynomials.

Definition 2.4.1 (S-polynomial)**.** *Let* $f \neq 0, g \neq 0 \in A_n$*. The S-polynomial of* f *and* g *is defined as*

$$S(f,g) = \frac{x^\gamma}{\mathrm{lt}(f)} \cdot f - \frac{x^\gamma}{\mathrm{lt}(g)} \cdot g$$

where $x^\gamma = \mathrm{lcm}(\mathrm{lm}(f), \mathrm{lm}(g))$

The S-polynomial of two polynomials is constructed in such a way that their leading terms cancel. The following theorem gives us an algorithmic test to check whether a set of polynomials constitutes a Gröbner basis.

Theorem 2.4.2 (Buchberger's criterion)**.** *Let* I *be an ideal in* A_n*. A finite subset* $G = \{g_1, \ldots, g_t\} \subset I$ *is a Gröbner basis if and only if for all pairs* $i \neq j$ *we have that* $S(g_i, g_j)$ *reduces to 0 with respect to* G*.*

Proof. See [CLO15] chapter 6, §6, theorem 6. □

input : $F = \{f_1, \ldots, f_m\}$ a sequence of polynomials and a monomial order $\leq$
output: A Gröbner basis for $\langle F \rangle$ with respect to $\leq$
begin
  $G := F$;
  $P :=$ `Sort(`$\{(p,q) : p, q \in G, p \neq q\}$`)` ;
  while $P \neq \emptyset$ **do**
    $(p,q) :=$ the first element of P ;
    $P := P \setminus \{(p,q)\}$;
    $r := S(p,q) \operatorname{rem} G$;
    if $r \neq 0$ **then**
      $P :=$ `Sort(`$P \cup \{(g,r) : g \in G\}$`)` ;
      $G := G \cup \{r\}$;
    end
  end
  return G
end

Algorithm 2: Buchberger's algorithm

Theorem 2.4.3. *Buchberger's algorithm 2 terminates in a finite number of steps and outputs a Gröbner basis for* $\langle f_1, \ldots, f_m \rangle$.

Proof. Write $I = \langle F \rangle$. We first prove correctness. We want to use theorem 2.4.2 to prove this. To this end, we need to show two things: that G is a subset of I during the entire execution of the algorithm and that at the end $S(g,h) \operatorname{rem} G = 0$ for all $g, h \in G$ with $g \neq h$. Now, at the start of the algorithm $G = F \subseteq I$. During each iteration of the while loop G is augmented with r. Since $p, q \in I$ it follows that $S(p,q) \in I$ and since $G \subseteq I$ we deduce that $r \in I$. Therefore, $G \cup \{r\} \subseteq I$. Whenever we process a new pair, if the remainder of the S-polynomial by G didn't already equal zero, then we add the remainder. This ensures that subsequent computation of the remainder will yield zero. Hence if the algorithm terminates we have that $S(g,h) \operatorname{rem} G = 0$ for all $g, h \in G$ with $g \neq h$.

Next, we prove that the algorithm indeed terminates. Every time a nonzero remainder is added to G the ideal $\langle \operatorname{lt}(G) \rangle$ strictly increases. This leads to an ascending chain of ideals in A_n. By Noetherianity of A_n this chain eventually stabilizes. This means that eventually the if-branch is never executed. Therefore the set P eventually becomes empty and the algorithm terminates. □

Unfortunately, the algorithm given above is not very efficient. Every time an S-polynomial is reduced to zero we don't gain any new information. As the reduction step is the most time consuming step it is natural to consider strategies avoiding these useless reductions. Since it is impossible to avoid reducing S-polynomials altogether, much effort has also gone in speeding up the reduction step. Finally, the order in which critical pairs are processed also has an effect on performance. This becomes especially important when the input polynomials are comprised of terms having different degree.
Buchberger gave two criteria to avoid useless reductions to zero.

Proposition 2.4.4 (Product criterion)**.** *Let $f, g \in A_n$ and assume that*

$$\gcd(\mathrm{lm}(f), \mathrm{lm}(g)) = 1.$$

Then $S(f, g)$ reduces to 0 with respect to $\{f, g\}$.

Proof. See [BWK93], lemma 5.66. □

Definition 2.4.5 (t-representation)**.** *Let $G = \{g_1, \ldots, g_s\} \in A_n$ be a set of polynomials, let t be a monomial, and let $f \in A_n$ be given. We say that f has a t-representation with respect to G if f can be written as*

$$f = p_1 g_1 + \ldots + p_s g_s$$

where we require that $p_i g_i < t$ whenever $p_i g_i \neq 0$. We will write that $f = O_G(t)$.

Proposition 2.4.6 (Chain criterion)**.** *Let F be a finite subset of A_n and $g_1, g_2, p \in A_n$ such that the following hold:*

- $\mathrm{lm}(p)$ *divides* $\mathrm{lcm}(\mathrm{lm}(g_1), \mathrm{lm}(g_2))$ *and*
- *$S(g_i, p)$ has a t_i-representation with respect to F with $t_i < \mathrm{lcm}(\mathrm{lm}(g_i), \mathrm{lm}(p))$ for $i = 1, 2$.*

Then $S(g_1, g_2)$ reduces to 0 with respect to F.

Proof. See [BWK93], proposition 5.70. □

2.5 The algebra of solving equations

In the area of error-correcting codes, the subject of the second part of this text, we will often need to solve a system of polynomial equations. In this section, we therefore discuss the relation between solving systems of polynomial equations and Gröbner bases. First we formalize what it means to solve a system of equations.

Definition 2.5.1 (Affine space)**.** *Given a field k and a positive integer n we define the n-dimensional affine space over k to be the set*

$$\mathbb{A}^n(k) = k^n = \{(a_1, \ldots, a_n) : a_1, \ldots, a_n \in k\}$$

It is convenient to think of the elements of A_n as functions on $\mathbb{A}^n(k)$. Indeed, every polynomial $f \in A_n$ defines a homomorphism $\mathbb{A}^n(k) \to k$ given by $(a_1, \ldots, a_n) \mapsto f(a_1, \ldots, a_n)$, the evaluation homomorphism. An element a of $\mathbb{A}^n(k)$ is called a (k-rational) point and the a_i its coordinates. A point $(a_1, \ldots, a_n)$ is called a zero of $f \in A_n$ if $f(a_1, \ldots, a_n) = 0$. We

will also refer to a zero of f as a solution to the equation $f = 0$. In other words, solving the equation $f = 0$ means finding a point $(a_1, \ldots, a_n) \in \mathbb{A}^n(k)$ such that $f(a_1, \ldots, a_n) = 0$. Consider $F = \{f_1, \ldots, f_m\}$, a set of polynomials in the polynomial ring A_n. Associated with F is a system of equations:

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_m(x_1, \ldots, x_n) &= 0 \end{aligned}$$

We are interested in the simultaneous solutions to these equations, i.e., the points $(a_1, \ldots, a_n) \in \mathbb{A}^n(k)$ solving each equation. These solutions form a geometric object, e.g., a single point in the case of a single solution. We call the geometric object an algebraic set.

Definition 2.5.2 (Algebraic set)**.** *Let F be a set of polynomials in A_n. We associate with F its locus of zeros,*

$$V_k(F) = \{(a_1, \ldots, a_n) \in \mathbb{A}^n(k) : f(a_1, \ldots, a_n) = 0 \text{ for all } f \in F\}$$

We will also write $V_k(f_1, \ldots, f_m)$ instead of $V_k(\{f_1, \ldots, f_m\})$. We may omit the subscript k when the context allows us to. A subset $X \subseteq \mathbb{A}^n(k)$ is called an algebraic set if $X = V(F)$ for some F.

If we write $I = \langle F \rangle$ for the ideal generated by the set of polynomials F, then it is not difficult to see that $V(I) = V(F)$. Hence, from now on we will only consider the ideal generated by a set of polynomials. By Hilbert's basis theorem every $X \subseteq \mathbb{A}^n(k)$ is the solution set of a finite number of equations. Moreover, going from ideals to the associated algebraic set and vice versa reverses inclusions: $I \subseteq J$ if and only if $V(J) \subseteq V(J)$.

Proposition 2.5.3. *Let $f_1, \ldots, f_m$ and $g_1, \ldots, g_s$ be sequences of polynomials in A_n such that $\langle f_1, \ldots, f_m \rangle = \langle g_1, \ldots, g_s \rangle$. Then the common zeros of the polynomials in $f_1, \ldots, f_m$ are the same as the common zeros of the polynomials in $g_1, \ldots, g_s$, i.e.,*

$$V_k(f_1, \ldots, f_m) = V_k(g_1, \ldots, g_s)$$

In particular we are interested in the case that $\{g_1, \ldots, g_s\}$ is a Gröbner basis for $\langle f_1, \ldots, f_m \rangle$.

Proof. Let $g \in \langle g_1, \ldots, g_s \rangle$, then $g \in \langle f_1, \ldots, f_m \rangle$, hence $g = u_1 f_1 + \cdots + u_m f_m$ for certain $u_1, \ldots, u_m \in A_n$. Now, let $a \in V_k(f_1, \ldots, f_m)$, then $g(a) = u_1(a) f_1(a) + \cdots + u_m(a) f_m(a) = 0$, so $a \in V_k(g_1, \ldots, g_s)$. The other direction goes analogous. □

Definition 2.5.4 (Vanishing ideal)**.** *Let $V \subseteq \mathbb{A}^n(k)$ be an algebraic set. Then $I(V) = \{f \in A_n : f(a) = 0 \text{ for all } a \in V\}$ is called the vanishing ideal of V.*

Observe that if $X = V(f_1, \ldots, f_m)$, then $\langle f_1, \ldots, f_m \rangle \subset I(X)$.

Definition 2.5.5 (Coordinate ring)**.** *Let $V \subseteq \mathbb{A}^n(k)$ be an algebraic set. The coordinate ring of V is the quotient ring $k[V] := A_n / I(V)$.*

The coordinate ring can be identified with an algebra of functions $V \to k$.

Theorem 2.5.6 (Weak Nullstellensatz)**.** *Let k be an algebraically closed field. Let I be a proper ideal in A_n. Then $V(I) \neq \emptyset$.*

Proof. See [Ful] chapter 1, §7. □

The intuition behind the weak Nullstellensatz is this: a system of equations $f_1 = f_2 = \cdots = f_m = 0$ is inconsistent if one is able to derive a contradictory statement such as $1 = 0$. This is equivelent with the existence of polynomials $q_1, \ldots, q_m$ such that $q_1 f_1 + \cdots + q_m f_m = 1$. So the only obstructions to solvability are the obvious ones.

Corollary 2.5.7. *Let k be an algebraically closed field. Let I be an ideal of the polynomial ring A_n. $V(I) = \emptyset$ if and only if $1 \in I$.*

Proof. Let $1 \in I$, then $V(I) = \emptyset$, because 1 never evaluates to zero. Conversely, assume that $V(I) = \emptyset$. Then $I = A_n$, by the weak Nullstellensatz. It follows that $1 \in I$. □

This gives us an algorithmic tool to test whether a system of equations has a solution. We simply compute a reduced Gröbner basis for I and check whether it equals $\{1\}$.

Definition 2.5.8 (Radical of an ideal)**.** *The radical of an ideal I in A_n, denoted $\sqrt{I}$, is defined by*

$$\sqrt{I} = \{f \in A_n : f^k \in I \text{ for some integer } k > 0\}$$

An ideal I such that $I = \sqrt{I}$ is called radical.

Theorem 2.5.9 (Hilbert's Nullstellensatz)**.** *Let k be an algebraically closed field. Let I be an ideal in A_n. Then $I(V(I)) = \sqrt{I}$.*

Proof. See [Ful] chapter 1, §7. □

This is saying that if f vanishes on $V(f_1, \ldots, f_m)$ there exist polynomials $q_1, \ldots, q_m$ such that $q_1 f_1 + \cdots + q_m f_m = f^k$ for some $k > 0$. Now, suppose that we are trying to solve a system of equations $f_1 = f_2 = \cdots = f_m = 0$ under the condition that $f \neq 0$. Then by the above this system has a solution, unless there exist polynomials $q_1, \ldots, q_m$ such that $q_1 f_1 + \cdots + q_m f_m = f^k$ for some $k > 0$.

Clearly Hilbert's Nullstellensatz implies the weak Nullstellensatz. Indeed, let $V(I) = \emptyset$, then $\sqrt{I} = I(V(I)) = I(\emptyset) = A_n$. From this we deduce that $1 \in \sqrt{I}$, and so $1 \in I$.
A consequence of Hilbert's Nullstellensatz is that over algebraically closed k there is a one-to-one correspondence between algebraic sets and radical ideals in A_n.

Definition 2.5.10 ((Affine) dimension of an ideal)**.** *Let I be an ideal in A_n. We define the affine dimension of I to be equal to the Krull dimension of A_n/I, i.e., the length of the longest chain of prime ideals in A_n/I. We denote this quantity as* $\dim I$.

Remark 1. This is the definition as given in [Eis95]. The motivation behind this is that if A_n/I is the coordinate ring of an algebraic set X, then the dimension of I corresponds to the geometric dimension of X. So we can define $\dim X = \dim I(X)$, although we won't explicitly use this in this thesis.

The systems of equations we consider will often have a finite number of solutions.

Theorem 2.5.11 (Finiteness Theorem)**.** *Let I be an ideal in A_n. Consider the following statements:*

1. *I is zero-dimensional.*
2. *$I \cap k[x_i] \neq \langle 0 \rangle$ for $1 \leq i \leq n$.*
3. *For each $1 \leq i \leq n$ there exists an $e_i \geq 0$ such that $x_i^{e_i} \in \langle \mathrm{lt}(I) \rangle$.*
4. *Let G be a Gröbner basis for I. For each $1 \leq i \leq n$ there exists an $e_i \geq 0$ such that $x_i^{e_i} = \mathrm{lm}(g)$ for some $g \in G$.*
5. *The k-vector space dimension $\dim_k A_n/I$ is finite.*
6. *The set $\{x^\alpha : x^\alpha \notin \langle \mathrm{lt}(I) \rangle\}$ is finite.*
7. *$V_k(I)$ consists of finitely many points.*

The statements 1 to 6 are equivalent and they all imply 7. If k is algebraically closed, then they are all equivalent.

Proof. See [CLO15] chapter 5, §3, theorem 6. □

Suppose now that we wish to check whether the system of equations defined by I has a finite number of solutions over the algebraic closure. By computing a Gröbner basis for I we can readily check statement 6, i.e., that the number of standard monomials is finite.

Definition 2.5.12 (Multiplicity of I)**.** *Let I be a zero-dimensional ideal in A_n. The Finiteness Theorem 2.5.11 implies that $\dim_k A_n/I$ is finite. We will call this quantity the multiplicity of I and denote it $\mathrm{mult}(I)$.*

We will later see that multiplicity is actually defined in a more general setting, and that it is related to the cardinality of $V(I)$.

Definition 2.5.13 (Perfect field)**.** *A field k is called perfect if either the characteristic of k equals 0 or the characteristic of k equals $p > 0$ and $k = k^p$.*

Example 2.5.14. *If $k = \mathbb{F}_q$ is any finite field, then k is perfect. Algebraically closed fields form another class of examples.*

Proposition 2.5.15. *Let I be a zero-dimensional ideal in A_n. Then $|V_{\overline{k}}(I)| \leq \mathrm{mult}(I)$. Moreover, if k is perfect and I is radical, then equality holds.*

Proof. See [KR08] theorem 3.7.19. □

Theorem 2.5.16 (Elimination theorem)**.** *Let G be a Gröbner basis for the ideal I in A_n with respect to the lexicographic order, where $x_1 > x_2 > \cdots > x_n$. For $j = 0, \ldots, n-1$ let $G_j = G \cap k[x_{j+1}, \ldots, x_n]$. This ideal is called the jth elimination ideal. Then G_j is a Gröbner basis for the jth elimination ideal.*

Proof. See [CLO15] chapter 3, §1, theorem 2. □

Remark 2. Recall that the lexicographic order is an elimination order. The theorem holds for general elimination orders as well, but we will only use the lexicographic order in this thesis for the purpose of elimination.

Proposition 2.5.17 (Triangular form)**.** *Let k be algebraically closed and let F be a system of polynomials in A_n and let G be the reduced Gröbner basis for $I = \langle F \rangle$ with respect to the lexicographic order with $x_1 > x_2 > \cdots > x_n$. If I is zero-dimensional, then G is in triangular form, i.e.,*

$$\begin{aligned} &x_n^{e_n} - g_n(x_n), \\ &x_{n-1}^{e_{n-1}} - g_{n-1}(x_{n-1}, x_n), \\ &\quad\vdots \\ &x_1^{e_1} - g_1(x_1, \ldots, x_{n-1}, x_n) \end{aligned}$$

where $e_i \geq 1$.

Proof. This follows immediately from theorem 2.5.11, the fact that G is a reduced Gröbner basis, and properties of the lexicographic order. □

Proposition 2.5.18. *Let k be an algebraically closed field and let I be an ideal in A_n. Then $V_k(I) = V_k(\sqrt{I})$.*

Proof. By Hilbert's Nullstellensatz we have that $I(V(I)) = \sqrt{I}$, hence $V(I) = V(I(V(I))) = V(\sqrt{I})$. □

Definition 2.5.19 (Squarefree polynomial)**.** *A non-constant polynomial f in $k[x]$ is called squarefree if there does not exist a non-constant $g \in k[x]$ such that g^2 divides f.*

Fortunately there exists an easy algorithm for checking whether a polynomial is squarefree, based on the following proposition.

Proposition 2.5.20. *A non-constant polynomial $f \in k[x]$ is squarefree if and only if* $\gcd(f, f') = 1$.

Theorem 2.5.21 (Seidenberg)**.** *Let k be a field and let I be a zero-dimensional ideal in A_n. Suppose that, for every $i \in \{1, \ldots, n\}$ there exists a squarefree polynomial in $I \cap k[x_i]$. Then $I = \sqrt{I}$.*

Proof. See [KR08] proposition 3.7.15. □

Corollary 2.5.22. *Let I be a zero-dimensional ideal and let f_i be such that $\langle f_i \rangle = I \cap k[x_i]$ for $1 \leq i \leq n$. Write g_i for the squarefree part of f_i. Then $\sqrt{I} = I + \langle g_1, \ldots, g_n \rangle$.*

Proof. See [KR08] corollary 3.7.16. □

Definition 2.5.23 (Normal position)**.** *Suppose that I is a zero-dimensional ideal in A_n. Let $i \in \{1, \ldots, n\}$. We say that I is in normal x_i-position, if any two zeros $a, b \in \mathbb{A}^n(\overline{k})$ of I satisfy $a_i \neq b_i$.*

Theorem 2.5.24. *Let I be a zero-dimensional radical ideal in A_n and assume that k is perfect. Finally, let g_n be the monic generator of $I \cap k[x_n]$. The following are equivalent:*

- *The ideal I is in normal x_n-position.*
- *The degree of g_n is equal to* $\mathrm{mult}(I)$.

Proof. See [KR08] theorem 3.7.23. □

Proposition 2.5.25 (Shape lemma)**.** *Let k be a perfect field and let F be a system of polynomials in A_n. Assume that $I = \langle F \rangle$ is zero-dimensional, radical, and in normal x_n-position. Let G be the reduced Gröbner basis for I with respect to the lexicographic order with $x_1 > x_2 > \cdots > x_n$. Let $|V(I)| = l$, then G has the form*

$$\{x_1 - g_1(x_n), x_2 - g_2(x_n), \ldots, x_{n-1} - g_{n-1}(x_n), x_n^l - g_n(x_n)\}$$

where every g_i is a univariate polynomial of degree at most $l-1$. In particular, $V_k(I) = \{(g_1(a_i), \ldots, g_{n-1}(a_i), a_i) : 1 \leq i \leq l\}$ where $a_1, \ldots, a_l$ are the roots of $x_n^l - g_n(x_n)$.

Proof. See [KR08] theorem 3.7.25. □

It is apparant that we need a method of finding the roots of a univariate polynomial. Substituting these roots into the g_i will yield $V(I)$.
If $(u_1, \ldots, u_n)$ is the unique solution to a system of equations, then the reduced Gröbner basis equals $\{x_1 - u_1, \ldots, x_n - u_n\}$. If the system has no solutions at all, then the reduced Gröbner basis equals $\{1\}$

2.5.1 The finite field case

Let $I \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ be an ideal of polynomials over $\mathbb{F}_q$. Recall that every element $a \in \mathbb{F}_q$ satisfies $a^q = a$. Now, write $I_q = I + \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle$. Suppose we were only interested in the zeros of I that lie in $\mathbb{F}_q$. Then

Proposition 2.5.26.

$$V_{\overline{\mathbb{F}}_q}(I_q) = V_{\mathbb{F}_q}(I)$$

Proof. Let $a \in V_{\overline{\mathbb{F}}_q}(I_q)$, then $f(a) = 0$ for all $f \in I_q$. In particular, if $f \in I \subseteq I_q$, then $f(a) = 0$. Moreover, we have that $a_i^q = a_i$ for $i = 1, \ldots, n$. It follows that $a_i \in \mathbb{F}_q$. Hence $a \in V_{\mathbb{F}_q}(I)$. The reverse inclusion is trivially true. □

Since I_q contains the polynomials $x_i^q - x_i$ for all $1 \leq i \leq n$, theorem 2.5.11 implies that I_q is zero-dimensional. Moreover, $x_i^q - x_i$ is squarefree, hence theorem 2.5.21 says that I_q is radical. Since $\overline{\mathbb{F}}_q$ is perfect, we only need to make sure that I_q is in normal x_n-position before the shape lemma 2.5.25 is applicable.

Proposition 2.5.27.

$$x^q - x = \prod_{a \in \mathbb{F}_q} (x - a)$$

Proof. Let $a \in \mathbb{F}_q$. The generalized Fermat theorem says that $a^q = a$. Hence, a is a root of $x^q - x$ and so $x - a$ divides $x^q - x$. It follows that $\prod_{a \in \mathbb{F}_q}(x - a)$ divides $x^q - x$. Since the degree of the divisor equals q it follows that they are, in fact, equal. □

An immediate consequence of this proposition is an application to root finding. Suppose that we are interested in the roots of a univariate polynomial $f \in \mathbb{F}_q[x]$. Calculating $\gcd(x^q - x, f)$ will give us the product of all the linear factors of f. The problem of finding the roots of f reduces to that of finding these factors. An application of the equal-degree factorization 3 by Cantor and Zassenhaus will yield these. We will describe only the method for odd values of q.

input : A squarefree monic polyomial $f \in \mathbb{F}_q[x]$ of degree $n > 0$, where q is assumed to be odd, and a divisor $d < n$ of n, so that all irreducible factors of f have degree d
output: A proper monic factor $g \in \mathbb{F}_q[x]$ of f, or "failure"
begin
  Choose $a \in \mathbb{F}_q[x]$ with $\deg a < n$ at random ;
  if $a \in \mathbb{F}_q$ **then**
    return *"failure"*
  end
  $g_1 := \gcd(a, f)$ **if** $g_1 \neq 1$ **then**
    return g_1
  end
  Compute $b = a^{\frac{q^d-1}{2}} \operatorname{rem} f$;
  $g_2 := \gcd(b - 1, f)$;
  if $g_2 \neq 1 \wedge g_2 \neq f$ **then**
    return g_2
  else
    return *"failure"*
  end
end

Algorithm 3: Equal-degree factorization

input : A nonconstant polyomial $f \in \mathbb{F}_q[x]$.
output: The distinct roots of f in $\mathbb{F}_q$.
begin
  Compute $h := x^q \operatorname{rem} f$ by repeating squaring. ;
  $g := \gcd(h - x, f)$;
  $r := \deg(g)$;
  if $r = 0$ **then**
    return [ ]
  end
  Compute the linear factors $x - u_1, \ldots, x - u_r$ of g by calling equal-degree factorization. ;
  return $[u_1, \ldots, u_r]$
end

Algorithm 4: Root finding

Alternatively, we can find the zeros of a polynomial over a finite field by simply trying out all of them. The Chien search 5 accomplishes just this.

input : A polynomial $\sigma(z) = 1 + \sum_{i=1}^{t} \sigma_i z^i \in \mathbb{F}_q[x]$
output: The zeros of $\sigma(z)$
begin
  Let $\alpha \in \mathbb{F}_q^*$ be a primitive nth root of unity. ;
  $Z := \emptyset$;
  $Q_j := \sigma_j$, for $j = 1, \ldots, t$;
  for $i := 0$ **to** $q - 1$ **do**
    $Q_j := Q_j \alpha^i$;
    if $1 + Q_1 + \cdots + Q_t = 0$ **then**
      $Z := Z \cup \{\alpha^i\}$;
    end
  end
  return Z
end

Algorithm 5: Chien search

As we have seen the Gröbner basis for a zero-dimensional ideal with respect to the lexicographic order has a very convenient shape regarding the solving of systems of equations. Often it is far more efficient to compute a Gröbner basis with respect to a degree-reverse lexicographic order. In [FMLG89] the FGLM algorithm is presented and it was developed exactly for this purpose.

Proposition 2.5.28. *Let I be a zero-dimensional ideal in A_n. Let $>_1$ and $>_2$ be two monomial orders. Finally, let G be a Gröbner basis for I with respect to $>_1$. The FGLM algorithm computes a Gröbner basis G' of I with respect to $>_2$ from G and its complexity is $\mathcal{O}(n(\text{mult}(I))^3)$.*

Proof. See [FMLG89]. □

2.6 Some projective geometry

It turns out that it is natural to augment affine space with so-called points at infinity. These are exactly the points that are "missing" in affine space. For example, in affine space two distinct lines might not intersect at all (they are parallel) whereas in projective space they do.

Definition 2.6.1 (Projective space)**.**

$$\mathbb{P}^n(k) = (\mathbb{A}^{n+1}(k) \setminus \{(0, \ldots, 0)\})/\sim \quad \textit{where } (a_1, \ldots, a_{n+1}) \sim (\lambda a_1, \ldots, \lambda a_{n+1}) \textit{ for } \lambda \in k^*$$

This definition suggest to think of n-dimensional projective space as the set of lines through the origin in $(n+1)$-dimensional affine space.

The equivalence class corresponding to $(a_1, \ldots, a_{n+1})$ is called a projective point and is denoted by $(a_1 : \cdots : a_{n+1})$ to emphasize that we only case about the ratio. Another point of view is that

$$\mathbb{P}^n(k) = \mathbb{A}^n(k) \cup \mathbb{P}^{n-1}(k)$$

where $\mathbb{P}^0(k)$ is a single point. This second definition immediately makes it clear that the n-dimensional affine space is naturally embedded into n-dimensional projective space by sending the point $(a_1, \ldots, a_n)$ to the projective point $(a_1 : \cdots : a_n : 1)$.

Definition 2.6.2 (Homogeneous polynomial)**.** *A polynomial $f \in A_{n+1}$ is said to be homogeneous of degree d if all terms of f have degree equal to d, i.e., it has the following shape,*

$$f = \sum_{\substack{\alpha \in \mathbb{N}^n \\ |\alpha| = d}} c_\alpha x^\alpha.$$

Note that the function defined by f does not have a fixed value on a projective point. However, it does make sense to ask whether $f(a)$ is equal to 0 or not. Thus f gives a function from $\mathbb{P}^n(k) \to \{0, 1\}$ defined by $f(a) = 0$ if $f(a_1, \ldots, a_n) = 0$ and $f(a) = 1$ if $f(a_1, \ldots, a_n) \neq 0$.

Definition 2.6.3 (Homogeneous ideal)**.** *An ideal I in A_{n+1} is called homogeneous if it admits a set of generators consisting of homogeneous polynomials.*

Now, many of the concepts we have defined in the previous section have their counterpart in projective space.

Definition 2.6.4 (Projective algebraic set)**.** *Let F be a set of homogeneous polynomials in A_{n+1}. We associate with F its locus of zeros,*

$$V_{p,k}(F) = \{a \in \mathbb{P}^n(k) : f(a) = 0 \text{ for all } f \in F\}$$

We will also write $V_{p,k}(f_1, \ldots, f_m)$ instead of $V_{p,k}(\{f_1, \ldots, f_m\})$. We will omit the subscript k when the context allows us to. A subset $X \subseteq \mathbb{P}^n(k)$ is called a projective algebraic set if $X = V_p(F)$ for some F consisting of a finite number of homogeneous polynomials.

Definition 2.6.5 (Homogeneous vanishing ideal)**.** *Let $X \subseteq \mathbb{P}^n(k)$. Then the ideal $I(X) = \{f \in A_{n+1} : f(a) = 0 \text{ for all } a \in \mathbb{P}^n(k)\}$ is called the homogeneous vanishing ideal of X.*

Remark 3. As the name suggest, if k is an infinite field, then the ideal $I(X)$ is homogeneous, for all $X \subseteq \mathbb{P}^n(k)$.

Theorem 2.6.6 (Projective Nullstellensatz)**.** *Let I be a homogeneous ideal in A_{n+1} and assume that k is algebraicallt closed. Then*

1. *$V_p(I) = \emptyset$ if and only if there is an integer d such that I contains all homogeneous polynomials of degree $\geq d$.*
2. *If $V_p(I) \neq \emptyset$, then $I_p(V_p(I)) = \sqrt{I}$.*

Proof. See [Ful] page 46. □

If V is an algebraic set in $\mathbb{P}^n(k)$ we define the affine cone over V by

$$C_V = \{(a_1, \ldots, a_{n+1}) \in \mathbb{A}^{n+1}(k) : (a_1 : \cdots : a_{n+1}) \in V\} \cup \{(0, \ldots, 0)\} \subseteq \mathbb{A}^{n+1}(k)$$

Definition 2.6.7 ((Projective) dimension of a homogeneous ideal)**.** *We define the projective dimension of a homogeneous ideal I in A_{n+1} to be the affine dimension plus one, and we denote it by $\dim_p I$.*

Remark 4. If I is homogeneous and has affine dimension equal to zero, then we always have that $V_{\overline{k}}(I) = \{0\}$. Equivalently, the algebraic set $V_{p,\overline{k}}(I)$ is empty! We mention this because several authors (see e.g. [BFSY05]) have tried to set up a theory based on the assumption that the ideal they are considering is both homogeneous and has affine dimension equal to zero. In practice, we are rarely interested in solving systems of homogeneous equations which only have zero as a solution.

There is a natural way of making any inhomogeneous polynomial homogeneous by adjoining a new variable to the underlying polynomial ring. We call this operation homogenization.

Definition 2.6.8 (Homogenization)**.** *Let $f \in A_n$ be an inhomogeneous polynomial of degree d. Such a polynomial can be homogenized by introducing an extra variable y. The resulting polynomial, denoted $h(f)$, then is*

$$h(f(x_1, \ldots, x_n)) = y^d f\left(\frac{x_1}{y}, \ldots, \frac{x_n}{y}\right) \in A_n[y]$$

In applications we need a way of reversing the operation of homogenization. We can accomplish this by simply setting the new variable equal to 1. We call this operation dehomogenization.

Definition 2.6.9 (Dehomogenization)**.** *Let $f \in A_n[y]$ be a homogeneous polynomial. Then $a(f) = f(x_1, \ldots, x_n, 1) \in A_n$ is called the dehomogenization of f with respect to y.*

The operations of homogenization and dehomogenization are not exactly inverse to each other. We do have that $a(h(f)) = f$ for all $f \in A$. However, if we let $f \in A_n[y]$ be a homogeneous polynomial such that y^m is the largest power of y dividing f, then we have that $h(a(f)) = y^{-m}f$. Fortunately, we are only interested in the former order of operations so this poses no problem for us.

Definition 2.6.10 (Homogenized ideal)**.** *Let I be an ideal in A_n. The homogenization of I is the ideal $\widetilde{I} = \{h(f) : f \in I\}$.*

Remark 5. Let $I = \langle f_1, \ldots, f_m \rangle \subseteq A_n$. We always have that $\langle h(f_1), \ldots, h(f_m) \rangle \subseteq \widetilde{I}$, but in general $\widetilde{I}$ is strictly larger.

Definition 2.6.11 (Projective closure)**.** *Given an affine algebraic set $V \subseteq \mathbb{A}^n(k)$ we define its projective closure as $\overline{V} = V_p(\widetilde{I(V)}) \subseteq \mathbb{P}^n(k)$.*

2.7 Some algebraic tools

In the next few subsections we will present some analytical tools from commutative algebra for two reasons. First, they are interesting in their own right. Second, they are often used in studying the degree of regularity, which we will discuss in a later chapter.

2.7.1 Projective Hilbert series

Every polynomial in A_n can be written as a sum of its homogeneous parts in a unique way. Writing $A_{n,d} = k[x_1, \ldots, x_n]_d = \{f \in A_n : f \text{ is homogeneous, } \deg(f) = d\} \cup \{0\}$ for the k-vector space of homogeneous polynomials of degree d, we thus deduce that

$$A_n = \bigoplus_{d \in \mathbb{N}} A_{n,d}.$$

Moreover, it is not hard to see that $A_{n,d}A_{n,e} \subseteq A_{n,d+e}$, making A_n into a graded ring. If I is homogeneous, then it is a graded ideal, since

$$I = \bigoplus_{d\in\mathbb{N}} I_d.$$

It follows that the algebra A_n/I is also graded. It is natural to study A_n/I by studying its components. For this reason the Hilbert function was introduced.

Definition 2.7.1 (Hilbert function)**.** *Let I be a homogeneous ideal in A_n. We define the function*

$$h_{A_n/I} : \mathbb{N} \to \mathbb{N}, \quad d \mapsto \dim_k A_{n,d}/I_d$$

and call it the Hilbert function of A_n/I.

Since the space $A_{n,d}$ is spanned by all monomials of degree d it is not difficult to see that $\dim_k A_{n,d} = \binom{n+d-1}{n-1}$, the number of columns of $M_{d,m}$. Moreover, $\dim_k I_d$ is equal to $\operatorname{rk} M_{d,m}$. In order to study the Hilbert function we turn to its generating function.

Definition 2.7.2 (Hilbert series)**.** *The generating function*

$$H_{A_n/I}(t) = \sum_{d=0}^{\infty} h_{A_n/I}(d)t^d = \sum_{d=0}^{\infty} (\dim_k A_{n,d}/I_d)t^d \in \mathbb{Z}[[t]]$$

is called the Hilbert series of I

Theorem 2.7.3 (Hilbert-Serre)**.** *Let I be a homogeneous ideal in A_n. There exists a polynomial $Q \in \mathbb{Z}[t]$ and $\delta \in \mathbb{N}$ such that*

$$H_{A_n/I}(t) = \frac{Q(t)}{(1-t)^\delta}$$

with $Q(1) \neq 0$ (In other words, $1 - t$ is not a factor of Q).

Proof. See [AM69] theorem 11.1 and cancel the common factors. □

Corollary 2.7.4 (Hilbert polynomial, index of regularity)**.** *There exists a polynomial $P_{A_n/I} \in \mathbb{Q}[d]$ such that*

$$h_{A_n/I}(d) = P_{A_n/I}(d)$$

for d large enough. This polynomial is called the Hilbert polynomial of A_n/I and

$$i_{reg}(I) = \inf\{d_0 : h_{A_n/I}(d) = P_{A_n/I}(d) \text{ for } d \geq d_0\}$$

is called the index of regularity of I.

Proof. Write

$$Q(t) = \sum_{k=0}^{\infty} a_k t^k$$

with $a_k \in \mathbb{Z}$ and only finitely many nonzero a_k. Now,

$$\frac{1}{(1-t)^\delta} = \sum_{j=0}^{\infty} \binom{j+\delta-1}{\delta-1} t^j$$

It follows that

$$H_{A_n/I}(t) = \sum_{j=0}^{\infty}\sum_{k=0}^{\infty} a_k \binom{j+\delta-1}{\delta-1} t^{j+k}$$

Since $H_{A_n/I}(t) = \sum_{d=0}^{\infty} h_{A_n/I}(d)t^d$ we have

$$h_{A_n/I}(d) = \sum_{k\leq d} a_k \binom{d-k+\delta-1}{\delta-1}$$

Now, when $d \geq \max\{k : a_k \neq 0\} = \deg(Q)$ we have that $h_{A_n/I}(d)$ is a univariate polynomial in d, i.e.,

$$P_{A_n/I}(d) = \sum_{k=0}^{\infty} a_k \binom{d-k+\delta-1}{\delta-1}$$

□

Proposition 2.7.5 (Some invariants)**.** *Write* $H_{A_n/I}(t) = \frac{Q(t)}{(1-t)^\delta}$ *with* $Q(1) \neq 0$ *and* $0 \leq \delta \leq n$*. Then*

1. $\deg(P_{A_n/I}) = \delta - 1$.
2. $\dim I = \delta$.
3. $i_{reg}(I) = \deg(Q) - \delta + 1$
4. $\operatorname{mult} I = (\delta-1)! \cdot \operatorname{lc}(P_{A_n/I})$ *if* $\delta > 0$ *and* $\dim_k A_n/I$ *if* $d = 0$ *(This will actually be our definition).*

Proof.

1. Oserve that the degree of the polynomial $\binom{x}{k} = \frac{x(x-1)(x-2)\cdots(x-k+1)}{k!}$ is equal to k. Since we have $k = \delta - 1$ the result immediately follows.
2. Corollary 13.7 in [Eis95] says that $\dim I = \dim A_n/I = 1+\deg(P_{A/I})$. From 1, it follows that $\dim I = \delta$.

□

Remark 6. Let I be a homogeneous ideal. If A_n/I is the coordinate ring of some $X \subseteq (P)^n(k)$ and if this set has geometric dimension d, then the multiplicity of I corresponds with the degree of X, i.e., the maximal possible number of intersection points with a linear space L of dimension $n - d$. As an example, if X has projective dimension 0, then $\deg X = |X|$. Similarly, if I is inhomogeneous and A_n/I is the coordinate ring of $X \subseteq \mathbb{A}^n(k)$, then the multiplicity of I corresponds with the degree of the projective closure $\overline{X} \subseteq \mathbb{P}^n(k)$.

Theorem 2.7.6 (Macaulay's bound)**.**

$$i_{reg}(I) \leq 1 + \sum_{k=1}^{m} (\deg(f_k) - 1)$$

Proof. See [BFS15]. □

2.7.2 Affine Hilbert Series

Besides being graded, A_n/I (where I is no longer assumed to be homogeneous) is naturally filtered. Let $A_{n,\leq d} = \{f \in A_n : \deg(f) \leq d\}$, and let $I_{\leq d} = I \cap A_{n,\leq d} = \{f \in I : \deg(f) \leq d\}$. We will study A_n/I by studying $A_{n,\leq d}/I_{\leq d})$.

Definition 2.7.7 (Hilbert function)**.** *Let I be an ideal in A_n. We define the function*

$$h^a_{A_n/I} : \mathbb{N} \to \mathbb{N}, \quad d \mapsto \dim_k A_{n,\leq d}/I_{\leq d}$$

and call it the affine Hilbert function of A_n/I.

Since the space $A_{n,d}$ is spanned by all monomials of degree at most d it is not difficult to see that $\dim_k A_{n,\leq d} = \binom{n+d}{d}$, the number of columns of $M_{d,m}$. The affine Hilbert function counts the number of monomials not in I of degree at most d. In order to study the Hilbert function we turn to its generating function.

Definition 2.7.8 (Hilbert series)**.** *The generating function*

$$H^a_{A_n/I}(t) = \sum_{d=0}^{\infty} h^a_{A_n/I}(d)t^d = \sum_{d=0}^{\infty} (\dim_k A_{n,\leq d}/I_{\leq d})t^d \in \mathbb{Z}[[t]]$$

is called the affine Hilbert series of I

Theorem 2.7.9 (Hilbert-Serre)**.** *Let I be an ideal in A_n. There exists a polynomial $Q \in \mathbb{Z}[t]$ and $\delta \in \mathbb{N}$ such that*

$$H^a_{A_n/I}(t) = \frac{Q(t)}{(1-t)^\delta}$$

with $Q(1) \neq 0$ (In other words, $1 - t$ is not a factor of Q).

Proof. This follows from [KR05] proposition 5.6.12 and theorem 2.7.3. □

Corollary 2.7.10 (affine Hilbert polynomial, affine index of regularity)**.** *There exists a polynomial $P^a_{A_n/I} \in \mathbb{Q}[d]$ such that*

$$h^a_{A_n/I}(d) = P^a_{A_n/I}(d)$$

for d large enough. This polynomial is called the Hilbert polynomial of A_n/I and

$$i^a_{reg}(I) = \inf\{d_0 : h^a_{A_n/I}(d) = P^a_{A_n/I}(d) \text{ for } d \geq d_0\}$$

is called the index of regularity of I.

Proof. This follows from [KR05] proposition 5.6.12 and corollary 2.7.4. □

Proposition 2.7.11 (Invariants)**.** *Write $H^a_{A_n/I}(t) = \frac{Q(t)}{(1-t)^{\delta+1}}$ with $Q(1) \neq 0$ and $0 \leq \delta \leq n$. Then*

1. $\deg(P^a_{A_n/I}) = \delta$.
2. $\dim I = \delta$.
3. $i^a_{reg}(I) = \deg(Q) - \delta$.

4. $\text{mult}(I) = Q(1)$.

Proof. The first two statements follow from [KR05] theorem 5.6.36. □

The affine and projective Hilbert function and series are related in the following way:

Proposition 2.7.12. *Let I be a proper homogeneous ideal in A_{n+1}, then*

1. $h^a_{A_{n+1}/I}(d) = \sum_{i=0}^{d} h_{A_{n+1}/I}(i)$.
2. $H^a_{A_{n+1}/I}(t) = \frac{H_{A_{n+1}/I}(t)}{1-t}$.

Proof. See [KR05] proposition 5.6.8. □

The following proposition enables us to translate properties derived from one series into the properties of the same name associated with the other series:

Proposition 2.7.13. *Let I be a proper homogeneous ideal of A_{n+1}, then*

1. $\dim I = \deg(P^a_{A_{n+1}/I})$.
2. $i_{reg}(I) = i^a_{reg}(I) + 1$.
3. $P_{A_{n+1}/I}(d) = P^a_{A_{n+1}/I}(d) - P^a_{A_{n+1}/I}(d-1)$.
4. $\deg I = (\dim I)! \cdot \text{lc}(P^a_{A_{n+1}/I})$.

Proof. See [KR05] proposition 5.6.11. □

2.7.3 Regular sequences

Regular sequences turn out to be useful in the context of the Matrix-F5 algorithm, which we will discuss in a later chapter.

Definition 2.7.14 (Regular sequence)**.** *A sequence of polynomials $f_1, \ldots, f_m$ in A_n is called a regular sequence if it satisfies the following conditions:*

- $\langle f_1, \ldots, f_m \rangle \neq A_n$.
- *If $gf_i = 0$ in $A_n/\langle f_1, \ldots, f_{i-1} \rangle$, then $g = 0$ in $A_n/\langle f_1, \ldots, f_{i-1} \rangle$ for all $g \in A_n$ and $1 \leq i \leq m$.*

The condition in the definition can be restated as follows: suppose that $gf_i \in \langle f_1, \ldots, f_{i-1} \rangle$, then $g \in \langle f_1, \ldots, f_{i-1} \rangle$.

Example 2.7.15. *The sequence $(x_1, \ldots, x_n)$ is regular in A_n.*

The polynomials constituting a regular sequence are "independent" from each other in a specific sense: the only relations they satisfy are the trivial ones, i.e., $f_i f_j = f_j f_i$, and the relations generated by them. This is captured in the following important theorem.

Theorem 2.7.16. *Let $F = \{f_1, \ldots, f_m\}$ be a sequence of homogeneous polynomials in A_n and let $I = \langle f_1, \ldots, f_m \rangle$. The following assertions are equivalent:*

- *F is a regular sequence.*
- *The syzygy module of I is generated by the principal syzygies, i.e., syzygies of the form $-f_j e_i + f_i e_j$.*

Proof. Suppose that the principal syzygies generate the syzygy module of I. Let $g \in A_n$ and suppose that $gf_i = 0$. If $g = 0$, then the result follows immediately. So we may suppose that $g \neq 0$. There exist $h_1, \ldots, h_{i-1} \in A_n$ such that

$$gf_i + \sum_{j=1}^{i-1} h_j f_j = 0.$$

In other words, we have a syzygy $(h_1, \ldots, h_{i-1}, g, 0, \ldots, 0)$ on $f_1, \ldots, f_m$. By assumption the syzygy module is generated by principal syzygies, hence there exist $h_{j,k} \in A_n$ such that

$$(h_1, \ldots, h_{i-1}, g, 0, \ldots, 0) = \sum_{1 \leq j < k \leq m} h_{j,k}(-f_k e_j + f_j e_k)$$

It follows that there exists j and k with $j < k < i$ such that $g = h_{j,k} f_j \in \langle f_1, \ldots, f_{i-1} \rangle$, as required.

Next, suppose that F is a regular sequence. We will provement the statement by induction on m. If $m = 1$, then $F = \{f_1\}$ and since F is regular, $f_1 \neq 0$. Since A_n is an integral domain, it follows that the only syzygy is 0, which is principal. For the inductive step, suppose that s is a syzygy, then

$$\sum_{i=1}^{m} s_i f_i = 0$$

It follows that $s_m f_m \in \langle f_1, \ldots, f_{m-1} \rangle : \langle f_m \rangle$. Since F is regular we have that

$$\langle f_1, \ldots, f_{m-1} \rangle : \langle f_m \rangle = \langle f_1, \ldots, f_{m-1} \rangle.$$

From this we deduce that there exist $h_1, \ldots, h_{m-1}$ such that $s_m = \sum_{i=1}^{m-1} h_i f_i$ Now,

$$\begin{aligned} s &= \sum_{i=1}^{m} s_i e_i \\ &= \sum_{i=1}^{m-1} s_i e_i + \left(\sum_{i=1}^{m-1} h_i f_i \right) e_m \\ &= \sum_{i=1}^{m-1} (s_i + h_i f_m) e_i + \sum_{i=1}^{m-1} h_i (-f_m e_i + f_i e_m) \end{aligned}$$

The term on the left is a syzygy on $f_1, \ldots, f_{m-1}$, so by the induction hypothesis we have that $\sum_{i=1}^{m-1} (s_i + h_i f_m) e_i$ is in the module of principal syzygies, and the term on the right is in the module of principal syzygies as well. It follows that every syzygy can be written in terms of principal syzygies. □

Theorem 2.7.17. *We are assuming that k is algebraically closed. Let $F = (f_1, \ldots, f_m)$ be a sequence of homogeneous polynomials in A_n, and let $I = \langle F \rangle$. Then the following statements are equivalent:*

1. *$(f_1, \ldots, f_m)$ is a regular sequence.*
2. *$\dim I = n - m$.*
3. *The Hilbert series of A/I is equal to*

$$H_{A/I}(t) = \frac{\prod_{i=1}^{m}(1 - t^{\deg(f_i)})}{(1-t)^n}$$

4. *The index of regularity meets the Macaulay bound, i.e., $i_{reg}(I) = 1 + \sum_{k=1}^{m}(\deg(f_k) - 1)$.*

Proof. See [Bar04] proposition 1.7.4. In the proof of statement 4 the paper by Lazard [Laz83] is referenced, but we were unable to find the exact statement in there. Moreover, we also found a reference to [Giu84], but couldn't find the exact statement in there either. We suspect it is stated there in a mathematical language we are unfamiliar with. □

From the theorem it immediately follows that an overdetermined homogeneous system can never be regular. In the inhomogeneous case, a permutation of a regular sequence is not necessarily a regular sequence.

Example 2.7.18. *For a counterexample, consider the ring $k[x, y, z]$ and the sequence $x, y(1-x), z(1-x)$. This sequence is regular. However, the sequence $y(1-x), z(1-x), x$ is not.*

If the polynomials are homogeneous, things are much nicer.

Corollary 2.7.19. *If $f_1, \ldots, f_m$ is a homogeneous regular sequence, then any permutation of this sequence is again a homogeneous regular sequence.*

Proof. This follows immediately from theorem 2.7.17 statement 3 and the commutativity of A_n. □

Chapter 3

Signature-based Gröbner basis theory

In 2002, Faugère [Fau02] first introduced the famous F5 algorithm. It is currently one of the fastest known algorithms for computing Gröbner bases and much of its speed can be attributed to the idea of so-called signatures to avoid redundant computations. All of the currently fastest algorithms are based on this idea. Therefore much research has been devoted to these signature-based Gröbner basis algorithms.
In this chapter we will discuss the fundamental ideas behind such algorithms and describe a basic signature-based algorithm based on Buchberger's algorithm to show how the ideas come together. In the following chapters, we will describe two signature-based algorithms that have been sucessfully used in practice. The first one is a matrix version of F5. It showcases all of the important ideas underlying the general F5 algorithm. Moreover, the mathematical theory behind it is particularly nice. The second algorithm is a relatively new algorithm which matches Buchberger's algorithm in simplicity while not sacraficing any speed compared to F5.

3.1 The module perspective

3.1.1 Relations between the generators: syzygies

The setting is as follows. Let $I \subseteq A_n$ be an ideal generated by a number of polynomials, say $I = \langle f_1, \dots, f_m \rangle$. We want to find a Gröbner basis for I, but we want to avoid useless zero-reductions. The main idea is to associate with each polynomial a vector of polynomials. The intuition behind this vector is that it will keep track of the representation of the polynomial in terms of the input sequence of polynomials. Hence, we consider the free module R^m over R where we let $e_1, \dots, e_m$ be the standard basis. Recall that e_i denotes the canonical unit vector, i.e. $e_i = (0, \dots, 0, 1, 0, \dots, 0)$.

Definition 3.1.1 (Syzygy module)**.** *Let $H = \{(a_1, \dots, a_m) \in A_n^m : a_1 f_1 + \cdots + a_m f_m = 0\}$. Recall that we call $(a_1, \dots, a_m)$ a syzygy. Therefore, we call H the syzygy module of I.*

We have the following exact sequence

$$0 \longrightarrow H \xrightarrow{\iota} A_n^m \xrightarrow{\phi} I \longrightarrow 0$$

where ι is the natural embedding and ϕ is the mapping given by $\phi(a_1, \ldots, a_m) = a_1 f_1 + \cdots + a_m f_m$. If $\phi(a) = f$ we will say that a is a representation of f. If k is a syzygy, then $f(k) = 0$ and so $f(a + k) = f$. It follows that f can have many different representations.

Definition 3.1.2 (Trivial/principal syzygy)**.** *The syzygies of the form $-f_j e_i + f_i e_j$ for $1 \leq i < j \leq m$ are called principal or trivial syzygies..*

These syzygies are called trivial since they follow immediately from the commutativity of A_n. Indeed, $\phi(-f_j e_i + f_i e_j) = -f_j f_i + f_i f_j = 0$.

Remark 7. By the existence of principal syzygies, the map ϕ is never injective. It follows that every $f \in I$ can have many representations.

3.1.2 Monomial orders and Gröbner bases for modules

In this subsection we extend the notion of a monomial order on A_n to one on A_n^m. We will also give the definition of a Gröbner basis for submodules of A_n^m. This is relevant to the chapter on the GVW algorithm, since it simultaneously computes a Gröbner basis for the ideal and the syzygy module of the generators of this ideal.
First, we define a monomial in A_n^m to be an element of the form $x^\alpha e_i$ for some $\alpha \in \mathbb{N}^n$ and $1 \leq i \leq m$. Now, a monomial order on A_n^m is defined analogously to a monomial order on A_n.

Definition 3.1.3 (Monomial order)**.** *A monomial order on A_n^m is a relation, denoted by $>$, on the set of all monomials in A_n^m satisfying*

- *$>$ is a total order.*
- *$>$ is compatible with multiplication, i.e., if $a > b$ and $x^\gamma \in T_n$, then $ax^\gamma > bx^\gamma$.*
- *$>$ is a well-order, i.e., there are no infinitely decreasing sequences of monomials.*

Usually, one takes a monomial order on A_n and extends it to A_n^m. Some of the common and useful monomial order on A_n^m are

Definition 3.1.4 (Some examples)**.** *Let $>$ be any monomial order on A_n.*

- *(TOP-extension of $>$) $x^\alpha e_i >_{TOP} x^\beta e_j$ if and only if $x^\alpha > x^\beta$ or $x^\alpha = x^\beta$ and $i > j$.*
- *(POT-extension of $>$) $x^\alpha e_i >_{POT} x^\beta e_j$ if and only if $i > j$ or $i = j$ and $x^\alpha > x^\beta$.*

As can be deduced from the definition, TOP stands for term-over-position and POT for position-over-term.
The monomial order $>$ on A_n and monomial order $>$ on A_n^m are said to be compatible if $x^\alpha > x^\beta$ if and only if $x^\alpha e_i > x^\beta e_i$ for all $\alpha, \beta \in \mathbb{N}^n$ and $i = 1, \ldots, m$.
Given an order on the monomials in A_n^m, any nonzero vector of polynomials $a \in A_n^m$ can be written as a sum of monomial terms

$$a = \sum_{i=1}^{d} c_i b_i$$

with $c_i \neq 0$ for $i = 1, \ldots, d$ and $b_1 > b_2 > \cdots > b_d$.

Definition 3.1.5. *We define the leading coefficient, monomial, term, and signature-poly pair as follows:*

- *The leading coefficient of a is* $\mathrm{lc}_>(a) = \mathrm{lc}(a) = c_1$.
- *The leading monomial of a is* $\mathrm{lm}_>(a) = \mathrm{lm}(a) = b_1$.
- *The leading term, henceforth called the signature, of a is* $\sigma(a) = c_1 b_1$.
- *The signature-poly pair of a is* $(\sigma(a), \phi(a))$.

Now, with this terminology the following definitions are exactly like their ideal counterparts.

Definition 3.1.6. *The set of leading terms of M is* $\mathrm{lt}(M) = \{\mathrm{lt}(f) : f \in M\}$.

Definition 3.1.7. *Let $>$ be a monomial order. A finite subset $G = \{g_1, \ldots, g_t\}$ of a module M is said to be a Gröbner basis for M if $\langle \mathrm{lt}(M) \rangle = \langle \mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_t) \rangle$.*

3.2 Buchberger's algorithm using signatures

In this section, we will describe a variant of Buchberger's algorithm using signatures to improve upon the classical case. Recall that the idea behind Buchberger's algorithm is to start with a basis of your ideal - not necessarily a Gröbner basis - and compute S-polynomials which are reduced by the polynomials found so far. If an S-polynomial reduces to zero we have accomplished nothing. If an S-polynomial has a nonzero remainder, then we add this remainder to the list of polynomials found so far. We keep doing this until we end up with a Gröbner basis, which is ensured by both Buchberger's criterion and the Noetherianity of A_n. Now, how do signatures fit into this story? Let's look at an example.

Example 3.2.1. *Consider $f = x, g = x^2 - y, h = x^3 - y^2$ in $R := \mathbb{Q}[x, y]$ with $>$ equal to the degree reverse lexicographic ordering with $x > y$ and extended to the POT-order in R^3. Computing S-polynomials we find that*

$$S(f, g) = y, \quad S(f, h) = y^2, \quad S(g, h) = -xy + y^2$$

and taking the remainder with respect to the ordered triple (f, g, h) yields,

$$S(f, g)\,\mathrm{rem}(f, g, h) = y, \quad S(f, h)\,\mathrm{rem}(f, g, h) = y^2, \quad S(g, h)\,\mathrm{rem}(f, g, h) = y^2.$$

We see that both $S(f, h)$ and $S(g, h)$ reduce to y^2. An immediate consequence is that $S(g, h)$ would reduce to zero with respect to (f, g, h, r) where $r = S(f, h)\,\mathrm{rem}(f, g, h)$. In other words, if we had added r to the set of polynomials in a previous iteration we would get a useless reduction in the iteration which selects $S(g, h)$ as the S-polynomial to reduce. Observe that we can zoom out to the level of the module and write

$$S(f, h) = y^2 = (x^2, 0, -1) \cdot (f, g, h), \quad S(g, h) = y^2 = (0, x, -1) \cdot (f, g, h).$$

Let's look at the signature of these module elements.

$$\sigma(x^2, 0, -1) = \sigma(0, x, -1) = -e_3.$$

It turns out that the signature is equal, which is the reason why they reduce to the same thing and why we only need to consider one of these S-polynomials. This is the power of the signature approach.

We will now describe the theory behing a signature-based variant of Buchberger's algorithm. We start by defining what it means for a module element to be divisible by another module element with the restriction that the signature is not allowed to decrease.

Definition 3.2.2. *Let $a \in A_n^m$ and let t be a term of $\phi(a)$. We say that b σ-reduces a if there exists a monomial $u \in T_n$ and a constant $c \in k$ such that*

1. *$cu\,\mathrm{lm}(\phi(b)) = t$ and,*
2. *$\sigma(a) > \sigma(ub)$.*

The corresponding σ-reduction is given by $a - cub$. In case $\sigma(a) = \sigma(ub)$ we speak of a singular σ-reduction. Otherwise, the reduction is called regular.

In a way, we are reducing division to the A_n-case, but by taking the associated module into account we are able to make sure to only perform divisions of a certain kind, i.e., the ones that respect the signature. The reason for doing this will become clear soon enough. First, we will need some extra terminology.

Definition 3.2.3. *Let $a, k \in A_n^m$ and let H be a set of elements of A_n^m. We say that a is σ-reduced to k by H if there exists a finite sequence of σ-reductions taking a to k by elements of H. So $k = a - c_1 m_1 h_1 - \cdots - c_l m_l h_l$.*

Definition 3.2.4. *Let $a \in A_n^m$ and let H be a set of elements of A_n^m. We say that a σ-reduces to zero with respect to H if there exists a syzygy k such that a is σ-reduced to k by H.*

Note that we do not require k to be zero. Since k is a syzygy, translating to I by ϕ will yield zero. This justisfies saying that it reduces to zero. A Gröbner basis in the classical sense has its signature counterpart.

Definition 3.2.5 (Signature Gröbner basis in signature T)**.** *Let I be an ideal in A_n and let $G = \{g_1, \ldots, g_s\}$ be a subset of vectors of polynomials in A_n^m. Furthermore, assume that $\phi(g_i)$ is monic for $i = 1, \ldots, s$. Then G is said to be a Gröbner basis in signature T for I if every element a of A_n^m with $\sigma(a) = T$ σ-reduces to 0 with respect to G.*

Definition 3.2.6 (Signature Gröbner basis up to signature T)**.** *G is said to be a signature Gröbner basis up to signature T for I if it is a Gröbner basis in signature S for all $S < T$.*

Definition 3.2.7 (Signature Gröbner basis)**.** *G is said to be a signature Gröbner basis for I if it is a Gröbner basis in all signatures.*

It will come as no surprise that there exists an analogue to the famous Buchberger's criterion. Before we introduce it, we need the notion of an S-vector.

Definition 3.2.8. *Let $0 \neq a, b \in A_n^m$ correspond to monic $\phi(a), \phi(b) \in A_n$. The S-vector of a and b is given by*

$$S(a,b) = \frac{\mathrm{lcm}(\mathrm{lm}(\phi(a)), \mathrm{lm}(\phi(b)))}{\mathrm{lm}(\phi(a))} a - \frac{\mathrm{lcm}(\mathrm{lm}(\phi(a)), \mathrm{lm}(\phi(b)))}{\mathrm{lm}(\phi(b))} b$$

When $\mathrm{lt}\left(\frac{\mathrm{lcm}(\mathrm{lm}(\phi(a)),\mathrm{lm}(\phi(b)))}{\mathrm{lm}(\phi(a))} a\right) = \mathrm{lt}\left(\frac{\mathrm{lcm}(\mathrm{lm}(\phi(a)),\mathrm{lm}(\phi(b)))}{\mathrm{lm}(\phi(b))} b\right)$ we call the S-vector singular. Otherwise, we call it regular.

Remark 8. Observe that $\phi(S(a,b)) = S(\phi(a),\phi(b))$.

Proposition 3.2.9 (Buchberger's criterion for signature Gröbner bases)**.** *Let I be an ideal in A_n and let $G = \{g_1,\ldots,g_s\}$ be a subset of vectors of polynomials in A_n^m. Furthermore, assume that $\phi(g_i)$ is monic for $i = 1,\ldots,m$. Then G is a signature Gröbner basis for I if and only if all $S(g_i,g_j)$ with $1 \leq i < j \leq m$ and all e_i with $1 \leq i \leq m$ σ-reduce to 0 with respect to G.*

Remark 9. The requirement that the e_i σ-reduce to zero with respect to G ensures that $\phi(G)$ is a basis for I and not for some ideal strictly contained in I.

Recall that we are interested in a Gröbner basis, but up to now we have defined everything in terms of signatures. Fortunately, it is easy to translate a signature Gröbner basis into a classical Gröbner basis.

Proposition 3.2.10. *If $G = \{g_1,\ldots,g_s\} \subseteq A_n^m$ is a signature Gröbner basis for I, then $\phi(G) = \{\phi(g_1),\ldots,\phi(g_s)\}$ is a Gröbner basis for I.*

Proof. Let $1 \leq i < j \leq s$ and consider the S-vector $S(g_i,g_j)$. Since G is a signature Gröbner basis, $S(g_i,g_j)$ σ-reduces to k where k is a syzygy. It follows that $\phi(S(g_i,g_j)) = S(\phi(g_i),\phi(g_j))$ reduces to $\phi(k) = 0$ (since k is a syzygy) with respect to $\phi(G)$. Hence $\phi(G)$ is a Gröbner basis for I. □

Proposition 3.2.11. *Let $\alpha,\beta \in A_n^m$ and let G be a signature Gröbner basis up to signature $\sigma(a) = \sigma(b)$. If a and b are both σ-reduced, then $\phi(a) = \phi(b)$.*

Proof. For the sake of contradiction, suppose that $\phi(a) \neq \phi(b)$. Since $\sigma(a) = \sigma(b)$ we deduce that $\sigma(a-b) < \sigma(a)$. Since G is a signature Gröbner basis up to signature $\sigma(a)$ we deduce that $a-b$ reduces to zero. W.l.o.g. we assume that $\mathrm{lt}(a-b)$ appears in a. But this contradicts that a was σ-reduced. □

Corollary 3.2.12. *When we process S-vectors in order of increasing signature, we need to process at most one S-vector at a given signature.*

When we σ-reduce an S-vector, say a, three things can happen.

1. a is a syzygy,
2. a is singular σ-reducible, or
3. we add a to the current basis.

The strength of the signature-based approach lies in a number of criteria for predicting zero reductions. First, we need some terminology.

Proposition 3.2.13. *Let $a \in A_n^m$ and G be a signature Gröbner basis up to $\sigma(a)$. If a is singular σ-reducible, then a σ-reduces to zero with respect to G.*

Proof. If a is singular σ-reducible then there exists a monomial $u \in T_n$ and a $g \in A_n^m$ such that $\sigma(ug) = \sigma(a)$. Let b be the result of reducing a by g. Then $\sigma(b) < \sigma(a)$. Hence b σ-reduces to 0 with respect to G. Hence a σ-reduces to 0 by $G \cup \{g\}$. □

Definition 3.2.14. *Let $a, b \in A_n^m$. We say that $\sigma(a)$ divides $\sigma(h)$ if there exists a monomial $m \in T_n$ and a constant $c \in k$ such that $\sigma(h) = cm(\sigma(a))$. Observe that this implies that both $\sigma(a)$ and $\sigma(h)$ contain the same basis vector e_i for some $1 \leq i \leq m$.*

Proposition 3.2.15 (Syzygy criterion)**.** *Let $G = \{g_1, \ldots, g_t\}$ be a set of elements of A_n^m and let $h = S(g_i, g_j)$. If G is a signature Gröbner basis up to signature $\sigma(h)$ and there exists a syzygy k such that $\sigma(k)$ divides $\sigma(h)$, then $S(g_i, g_j)$ σ-reduces to zero with respect to G.*

Proof. By assumption, there exists an $x^\alpha \in T_n$ such that $h = x^\alpha k$. Since k is a syzygy, we also have that $x^\alpha k$ is a syzygy and this syzygy has signature equal to $\sigma(h)$. It follows that $\sigma(h - x^\alpha k) < \sigma(h)$. Hence $\phi(h - x^\alpha k) = \phi(h) - x^\alpha \phi(k) = 0$. This implies that $\phi(h) = 0$. Equivalently, h σ-reduces to zero with respect to G. □

Definition 3.2.16 (Predictably syzygy)**.** *A signature T is said to be predictably syzygy if*

- *there exists a syzygy p that is generated by principal syzygies and is such that $\sigma(p) = T$, or*
- *there exists a syzygy $q \in A_n^m$ such that $\sigma(q) < T$ and $\sigma(q)$ divides T.*

input : $F = \{f_1, \ldots, f_m\}$ a sequence of polynomials, and a monomial order $\leq$ on A_n
output: A signature-Gröbner basis for $\langle F \rangle$ with respect to $\leq$
begin
  $G := \emptyset$;
  $P := \{e_1, \ldots, e_m\}$;
  $H := \{-f_j e_i + f_i e_j : 1 \leq i < j \leq m\}$;
  while $P \neq \emptyset$ **do**
    $p :=$ an element of P with $\leq$-minimal signature ;
    $P := P \setminus \{p\}$;
    if $\neg$ `Criterion`$(p, G \cup H)$ **then**
      $p' :=$ result of regular σ-reducing p by G ;
      if $\phi(p') = 0$ **then**
        $H := H \cup \{\sigma(p')\}$;
      else
        $p' := \frac{1}{\mathrm{lc}(\phi(p'))} p'$;
        $P := P \cup \{S(g, p') : g \in G \text{ and } S(g, p') \text{ is regular}\}$;
        $G := G \cup \{p'\}$;
      end
    end
  end
  return $\phi(G)$
end

Algorithm 6: Signature Buchberger's algorithm

Theorem 3.2.17. *Algorithm 6 terminates in a finite number of steps and outputs a Gröbner basis.*

Proof. See [EF15] theorem 5.2 and proposition 3.2.10. □

Chapter 4

Linearization and the Matrix-F5 algorithm

In this chapter, we will explore the link between the calculation of Gröbner bases and linear algebra. We present the Matrix-F5 algorithm [Bar04], our first example of a signature-based Gröbner basis algorithm, which uses the F5-criterion to predict useless reductions to zero and linear algebraic methods to speed up the reduction process. While interesting in its own right, Matrix-F5 is also used to study the complexity of the general F5-algorithm (See [BFS15]).

Suppose we are interested in computing a Gröbner basis for the ideal $I = \langle f_1, \ldots, f_m \rangle \subseteq A_n$. The key observation is that I is a k-vector space in addition to being an ideal. In what follows, we will therefore be careful in distinguishing between an ideal basis and a linear basis. We will emphasize the word linear when needed. Observe that I is linearly generated by the set

$$\{x^\alpha f_i : 1 \leq i \leq m, \alpha \in \mathbb{N}^n\}.$$

We can encode these generators into an infinite matrix. This matrix will contain a column for each monomial and a row for each product $x^\alpha f_i$. The entry indexed by such a pair is the coefficient of the monomial in the product. The idea is that a Gröbner basis for I can be derived from a triangulation of this matrix.

Remark 10. The fact that, in theory, we can triangulate this infinite matrix follows from the fact that each row consists of only a finite number of nonzero entries, since they correspond to polynomials. Moreover, each column contains a finite number of nonzero entries because it corresponds to a monomial of a fixed degree, say d, and this monomial appears only in the polynomials of the form $x^\alpha f_i$ with $|\alpha| + \deg(f_i) = d$. These are finite in number.

It is this connection that was first explored by Lazard in [Laz83], who in turn based his ideas on the work of Macaulay [Mac94].

4.1 The homogeneous case

We will first restrict our attention to the case where each f_i is a homogeneous polynomial, as the theory is much simpler in this case. So we are assuming that I is a homogeneous ideal. This assumption is not a restriction. As we will later see, any inhomogeneous polynomial can be made homogeneous by introducing a new variable. Homogeneous polynomials offer several

advantages over inhomogeneous polynomials. For example, when the monomials are ordered by a degree compatible ordering, the matrix as described above consists of blocks which do not overlap. In particular when we look at the submatrix comprised of the nonzero columns it is a direct sum of special matrices which we will encounter shortly. Recall that we write $A_{n,d} = k[x_1, \ldots, x_n]_d = \{f \in A_n : f \text{ is homogeneous, } \deg(f) = d\} \cup \{0\}$ for the k-vector space of homogeneous polynomials of degree d, and recall that A_n can be decomposed as

$$A_n = \bigoplus_{d \in \mathbb{N}} A_{n,d}.$$

Now, let $I_d = I \cap A_{n,d} = \{f \in I : f \text{ is homogeneous, } \deg(f) = d\} \cup \{0\}$. So

$$I = \bigoplus_{d \in \mathbb{N}} I_d$$

It follows that finding a linear basis for I (which, in general, is infinite-dimensional) is equivalent to finding a linear basis for I_d for each $d \geq 0$. It is not hard to see that I_d is linearly generated by the monomials

$$S_d := \{x^\alpha f_i : |\alpha| + \deg(f_i) = d, 1 \leq i \leq m\}$$

so we can explicitly list $\dim_k I_d$ basis vectors. Consider the canonical isomorphism between I_d and $k^{|T_{n,d}|}$ identifying a polynomial with its vector of coefficients with respect to the basis $T_{n,d}$. Here $|T_{n,d}| = \binom{n+d-1}{d}$, the number of monomials in degree d. We can collect the coefficient vectors associated with the polynomials in S_d into a matrix mentioned at the beginning of this section and call it a Macaulay matrix.

Definition 4.1.1 (Macaulay matrix)**.** *Fix a monomial order $>$. Given a sequence of homogeneous polynomials $f_1, \ldots, f_m \in A_n$ and a degree $d \in \mathbb{N}$ we can construct the Macaulay matrix $M_{d,m}$ having its columns indexed by $T_{n,d}$ in decreasing order. Multiply each f_i of degree d_i with each monomial $x^\alpha \in T_{n,d-d_i}$ in increasing order. The resulting matrix contains a row indexed by $x^\alpha f_i$. The value at position $(x^\alpha f_i, x^\beta)$ is given by the coefficient of x^β in $x^\alpha f_i$.*

$$M_{d,m} = \begin{array}{c} \\ \vdots \\ x^{\alpha_i} f_k \\ \vdots \end{array} \begin{array}{c} \ldots x^{\beta_j} \ldots \\ \\ \left(\textit{Coefficient}(x^{\beta_j}, x^{\alpha_i} f_k) \right) \\ \\ \end{array}$$

Example 4.1.2. *Let the ring be $\mathbb{F}_3[x, y, z]$ equipped with the degree reverse lexicographic order, and consider the polynomials*

$$f_1 = x + y + z \text{ and } f_2 = y + 2z$$

The Macaulay matrix of f_1 and f_2 in degree 1 is given by:

$$M_{1,2} = \begin{array}{c} \\ f_1 \\ f_2 \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix} \end{array}$$

This example shows that we simply write the polynomials in terms of the vector space basis. Now, since $z < y < x$ we obtain the following matrix in degree 2:

$$M_{2,2} = \begin{array}{c} \\ zf_1 \\ yf_1 \\ xf_1 \\ zf_2 \\ yf_2 \\ xf_2 \end{array} \begin{array}{c} \begin{array}{cccccc} x^2 & xy & xz & y^2 & yz & z^2 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 2 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

If we now compute an echelon form of $M_{d,m}$, say $\widetilde{M}_{d,m}$ (henceforth a tilde denotes an echelon form), then the polynomials associated with row vectors in this echelon form yield a linear basis for I_d. Computing an echelon form can be done by applying the Gaussian elimination algorithm to $M_{d,m}$. The resulting basis has a property reminiscent of those of a Gröbner basis. This inspires the following definition:

Definition 4.1.3 (Gröbner basis for degree d)**.** *Let I be a homogeneous ideal in A_n and let $d \in \mathbb{N}$. A subset G of I is called a Gröbner basis for I for degree d if for every $f \in I_d$ there exists a $g \in G$ such that $\operatorname{lt}(g)$ divides $\operatorname{lt}(f)$.*

Indeed,

Proposition 4.1.4. *The linear basis for I_d given by the rows of $\widetilde{M}_{d,m}$ is a Gröbner basis for I for degree d.*

Proof. Let $f \in I_d$. Write $\{b_1, \ldots, b_s\}$ for the ordered (b_i corresponds to the ith row) basis for I_d. Then $f = a_1 b_1 + \cdots a_s b_s$ with $a_i \in k$. Let j be the first index such that a_j is nonzero. It follows that $\operatorname{lt}(f) = \operatorname{lt}(a_1 b_1 + \cdots a_s b_s) = a_j \operatorname{lt}(b_j)$ by the properties of the echelon form. Hence $\operatorname{lt}(b_j)$ divides $\operatorname{lt}(f)$, as required. □

Let us generalize this. Write $I_{\leq d} = \{f \in I : \deg(f) \leq d\}$ for the ideal of all polynomials in I having degree at most d. Then $I_{\leq d} = \bigoplus_{k=0}^{d} I_k$ as a sum of linear spaces. A linear basis for this space is given by the union of the bases for the I_k. As we have just seen these are easy to compute. The linear basis for $I_{\leq d}$ forms what is called in the literature a d-Gröbner basis, see e.g. [FSEDS11] or [BFS15].

Definition 4.1.5 (Gröbner basis up to degree d / d-Gröbner basis / d-truncated Gröbner basis)**.** *Let I be a homogeneous ideal in A_n and let $d \in \mathbb{N}$. A subset G of I is called a Gröbner basis up to degree d of I, or d-Gröbner basis for short, if for every $f \in I_{\leq d}$ there exists a $g \in G$ such that $\operatorname{lt}(g)$ divides $\operatorname{lt}(f)$.*

The relation between the two definitions is captured by the following proposition:

Proposition 4.1.6. *Let G_d denote a Gröbner basis for I for degree d, then $G = \bigcup_{d=0}^{D} G_d$ is a Gröbner basis up to degree D.*

Proof. Let $f \in I_{\leq D}$ and let $d = \deg(f)$. Then $\operatorname{lt}(f) \in I_d$. Hence there exists a $g \in G_d \subseteq G$ such that $\operatorname{lt}(g)$ divides $\operatorname{lt}(f)$. Hence G is a Gröbner basis up to D. □

All the ideas described above can easily be incorporated into an algorithm. Algorithm 7 incrementally computes d-Gröbner bases until we end up with a D-Gröbner basis, where D is an input parameter. For some applications having a D-Gröbner basis is enough, e.g., when

input : $F = (f_1, \ldots, f_m)$ a sequence of polynomials, a positive integer D, and a monomial order $>$ on A_n
output: A D-Gröbner basis for $\langle F \rangle$ with respect to $>$
begin
  $G := [\,]$;
  for $d := 1$ **to** D **do**
    $M := [\,]$;
    for $j := 1$ **to** m **do**
      if $\deg(f_j) = d$ **then**
        `Append(`M, f_j`)` ;
      else if $\deg(f_j) < d$ **then**
        $M_{d-\deg f_j} :=$ all monomials of degree $d - \deg(f_j)$;
        for $t \in M_{d-\deg f_j}$ **do**
          `Append(`M, tf_j`)` ;
        end
      end
    end
    $\widetilde{M} :=$ `GaussianElimination(CoefficientMatrix(`M`))` ;
    $G := G \cup \{h \in \widetilde{M} : \forall g \in G : \mathrm{lm}(g) \text{ does not divide } \mathrm{lm}(h)\}$;
  end
  return G
end

Algorithm 7: Lazard's algorithm

wanting to test whether some f of degree at most D is in I. We are however interested in a Gröbner basis in the usual sense as our ultimate goal is system solving. Fortunately, for large enough D these two notions coincide. This is captured by the following theorem:

Theorem 4.1.7. *There exists a $d_\infty \in \mathbb{N}$ such that for every $d \geq d_\infty$ we have that if G is a Gröbner basis up to degree d it is also a Gröbner basis.*

Proof. Consider the following ascending chain of ideals in A_n:

$$\langle \mathrm{lm}(I_{\leq 0}) \rangle \subseteq \langle \mathrm{lm}(I_{\leq 1}) \rangle \subseteq \langle \mathrm{lm}(I_{\leq 2}) \rangle \subseteq \cdots$$

Since A_n is a Noetherian ring the chain eventually stabilizes, i.e., there exists a d^∞ such that $\langle \mathrm{lm}(I_{\leq d_\infty}) \rangle = \langle \mathrm{lm}(I_{\leq d_\infty+1}) \rangle = \langle \mathrm{lm}(I_{\leq d_\infty+2}) \rangle = \cdots$. Now, let G be a Gröbner basis up to degree d for some $d \geq d_\infty$, and let $f \in I$. Then $\mathrm{lm}(f) \in \langle \mathrm{lm}(I) \rangle$. Observe that $\langle \mathrm{lm}(I) \rangle = \langle \mathrm{lm}(I_{\leq d}) \rangle = \langle \mathrm{lm}(I_{\leq d_\infty}) \rangle$. Hence $\mathrm{lm}(f) \in \langle \mathrm{lm}(I_{\leq d}) \rangle$. It follows that there exists a monomial $u \in I_{\leq d}$ dividing $\mathrm{lm}(f)$. Since G is a Gröbner basis up to degree d there exists a $g \in G$ such that $\mathrm{lm}(g)$ divides u. By transitivity $\mathrm{lm}(g)$ divides $\mathrm{lm}(f)$. This implies that G is a Gröbner basis. □

Intermezzo.

We feel that the degree at which stabilization takes place deserves its own name:

Definition 4.1.8 (Degree of regularity)**.** *We define the degree of regularity as the smallest d_∞ satisfying the conditions in theorem 4.1.7 and denote it by $d_{reg}(I)$.*

Remark 11. In the literature, the degree of regularity is defined in a different way (see e.g. [BFSY05]). The problem with the definition the authors present in there is that it only make sense in the context of a homogeneous zero-dimensional (in the affine sense) ideal. As we remarked in the section on projective geometry, such ideals are not interesting at all in practice. We also want to point out that $d_{reg}(I)$ depends on the monomial order being used.

We quote a theorem by Giusti [Giu84] which seems helpful, although we do not fully understand its assumptions. It seems to say that this bound holds for "random" systems of equations.

Theorem 4.1.9. *Let $>$ be the lexicographic ordering. In the finite-dimensional vector space parametrizing the ideals of $k[x_1, \ldots, x_n]$ generated by k polynomials of degree less than d, there exists a non-empty Zariski-open subset where $d_{reg}(I)$ is bounded by $nd - n + 1$.*

Proof. See [Giu84] theorem C. □

Another useful theorem by Giusti is the following:

Theorem 4.1.10. *Let $I = \langle f_1, \ldots, f_m \rangle$ be a (not necessarily homogeneous) ideal in A_n and let $>$ be the degree reverse lexicographic ordering. Let $d = \max\{\deg(f_i) : 1 \leq i \leq m\}$. Let I^h denote the ideal generated by $h(f_1), \ldots, h(f_m)$ (Recall that h denotes the homogenization map) and assume that $\dim A_n/I^h = 1$. Then the highest degree appearing during the computation of a Gröbner basis is at most $1 + (n-1)(d-1)$. In other words, $d_{reg} \leq 1 + (n-1)(d-1)$.*

Proof. See [Giu84] theorem 3.9 on page 170. □

Remark 12. In practice one often encounters system of equations having degree at most 2. If the assumptions in the theorem are satisfied, then it says that $d_{reg} \leq n$ for such a system. In other words, d_{reg} would be bounded above by the number of variables.

Clearly d_{reg} says a lot about the complexity of computing a Gröbner basis and we feel that there is much more that can be said. See e.g. the paper by Ding et al. [DS13] which makes the observation that there seems to be a lot of confusion about this topic within the cryptographic community.

End of intermezzo.

Theorem 4.1.11. *Lazard's algorithm 7 terminates and computes a D-Gröbner basis for $\langle F \rangle$.*

Proof. Termination follows from the fact that the number of iterations of the outer loop is bounded by D and, similarly, the number of iterations of the inner loop is bounded by m. That the algorithm computes a D-Gröbner basis is a consequence of propositions 4.1.4 and 4.1.6. □

Corollary 4.1.12. *Let I be a homogeneous ideal. Then all polynomials in the reduced Gröbner basis G of I have degree at most $d_{reg}(I)$.*

Proof. When Lazard's algorithm terminates, it outputs a D-Gröbner basis. Elements in this D-Gröbner basis have degree at most D. When $D = d_{reg}(I)$ theorem 4.1.7 says that G is a Gröbner basis. In the reduced Gröbner basis of I the degree and number of polynomials can only drop, thus the bound follows. □

4.2 Using known linear dependencies

Even though algorithm 7 is correct and constructs a Gröbner basis for large enough D, it is not very efficient; it spends a lot of time doing redunant computations. We will now describe a series of impovements ultimately leading to a matrix variant of the F5 algorithm. We follow the exposition by Albrecht [Alb10]. The first improvement is best illustrated by a small example. To this end, let $>$ be the degree reverse lexicographic order and consider the following polynomials in $\mathbb{F}_3[x, y, z]$:

$$f_1 = x + y + z, f_2 = 2x + z, f_3 = 2y + z.$$

First, construct the Macaulay matrix in degree 1 associated with this system:

$$M_{1,3} = \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 1 & 1 & 1 \\ 2 & 0 & 1 \\ 0 & 2 & 1 \end{pmatrix} \end{array}$$

A row echelon form of $M_{1,3}$ is:

$$\widetilde{M}_{1,3} = \begin{array}{c} \\ f_1 \\ f_4 \\ f_5 \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

This gives us a new polynomial $f_4 = y + 2z$ in $\langle f_1, f_2, f_3 \rangle$. Now, in constructing $M_{2,3}$ we multiply f_i by z, y, and x (in that order) for $1 \leq i \leq 3$ and construct the corresponding rows. Observe that if we compute the row echelon form of the constructed matrix the work we do is redundant. Indeed, $f_4 = f_1 + f_2$ from which it follows that $xf_4 = xf_1 + xf_2$. So instead of reducing the row xf_2 by the row xf_1 in $M_{2,3}$ we simply replace xf_2 by xf_4, which we know from the previous step. Moreover, $f_5 = f_1 + f_2 + f_3 = 0$, so we say that f_3 reduced to zero. Any multiple of f_3 will reduce to zero as well. Hence we may leave out f_3 altogether. In general, let $f_j \neq 0$ be a newly discovered polynomial in the dth iteration of algorithm 7 obtained by reducing a row $x^\alpha f_i$, then we form the matrix $M_{d+1,m}$ using the polynomial f_j instead of $x^\alpha f_i$. In our example, this will lead to the following matrix:

$$M_{2,3} = \begin{array}{c} \\ zf_1 \\ yf_1 \\ xf_1 \\ zf_4 \\ yf_4 \\ xf_4 \end{array} \begin{array}{c} \begin{array}{cccccc} x^2 & xy & xz & y^2 & yz & z^2 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 2 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

This incremental strategy poses a new problem. Let $x^\alpha f_i$ be a row in $M_{d,m}$. Then in $M_{d+2,m}$ we will have both the rows $xyx^\alpha f_i$ and $yxx^\alpha f_i$. By commutativity these two rows are equal. We want to avoid this from happening. The idea of a signature, which we have seen in chapter 3, turns out to be the right tool to solve this problem. In the language of Faugère [Fau02], we first define the notion of a labeled polynomial.

Definition 4.2.1 (Labeled polynomial)**.** *Let $F = \{f_1, \ldots, f_m\}$ be a sequence of polynomials in A_n and let $f \in \langle F \rangle =: I$ be a polynomial. Recall that ϕ is the obvious homomorphism between A_n and I. Then there exists a vector $(h_1, \ldots, h_m) \in A_n^m$ such that $f = \phi(h_1, \ldots, h_m)$. We call $\sigma(f) = \sigma(h_1, \ldots, h_m)$ the signature of f and call $(\sigma(f), f)$ a labeled polynomial. Note that this corresponds with the notion of signature-poly pair as defined in chapter 3.*

The matrix version of F5 that we will introduce shortly uses the POT-extension of $>$ (Here $>$ is any monomial order given on input) for its signature ordering.

Example 4.2.2. *The polynomial $x^\alpha f_i$ has signature equal to $x^\alpha e_i$, so the corresponding labeled polynomial is $(x^\alpha e_i, x^\alpha f_i)$.*

Example 4.2.3. *The polynomial $x^\alpha f_i + x^\beta f_j$ where $i < j$ has signature equal to $x^\beta e_j$, so the corresponding labeled polynomial is $(x^\beta e_j, x^\alpha f_i + x^\beta f_j)$.*

If we multiply $x^\alpha f_i$ only by variables larger (with respect to the underlying monomial order) than the largest variable occurring in x^α then we will never be in the situation which we have just described. By the introduction of signatures a new problem arises. According to the theory of chapter 3 we only allow regular σ-reductions. However, Gaussian elimination with pivoting freely swaps rows and performs singular σ-reductions as well. We can modify it such that it does not allow row swaps and such that rows are only affected by rows preceding it. The resulting algorithm is algorithm 8.

Remark 13. While algorithm 8 is correct, it is not particularly fast. In practice one wants to modify existing fast linear algebra algorithms as computation of an echelon form is what determines the speed of the overall algorithm.

```
input  : An m × n matrix A
output: A row echelon form of A
begin
    for j := 1 to n do
        for i := 1 to m do
            if A[i, j] ≠ 0 then
                b := false ;
                for k := 1 to j − 1 do
                    if A[i, k] ≠ 0 then
                        b := true ;
                    end
                end
                if b then
                    continue;
                end
                Multiply the ith row by 1/A[i,j] ;
                for i := k + 1 to m do
                    if A[k, j] ≠ 0 then
                        Eliminate the entry A[k, j] using row i ;
                    end
                end
                break;
            end
        end
    end
    return A
end
```

Algorithm 8: Modified Gaussian elimination

Remark 14. The labeled polynomials occurring in the algorithm are always of the form $(x^\alpha e_i, x^\alpha f_i)$ for some monomial $x^\alpha \in T_n$. In most implementations it is more efficient to not use the vector representation of the signature, but rather the tuple (x^α, i). We will therefore use the latter and in the text below we will use both representations.

4.3 Predicting zero reductions

Recall that a principal syzygy is any syzygy of the form

$$-f_j e_i + f_i e_j$$

As an example, consider any element in the module generated by the principal syzygies. It can be written as

$$u(-f_2 e_1 + f_1 e_2) + v(-f_3 e_1 + f_1 e_3) + w(-f_3 e_2 + f_2 e_3)$$

where $u, v, w \in A_n$. This can be rewritten as

$$(vf_1 + wf_2)e_3 + (uf_1 - wf_3)e_2 + (-uf_2 - vf_3)e_1$$

Now, notice that $vf_1 + wf_2 \in \langle f_1, f_2 \rangle$. The converse holds as well: since v and w were arbitrary, any $h \in \langle f_1, f_2 \rangle$ gives rise to a number of syzygies in the module generated by the principle syzygies. Having computed a Gröbner basis G of $\langle f_1, f_2 \rangle$ it is easy to check whether h is in this ideal. In the case that h is a monomial, we only need to check whether h is divisible by $\mathrm{lt}(g)$ for some $g \in G$. This example easily generalizes to $f_1, \ldots, f_i$ and lies at the heart of the Matrix-F5 algorithm. Let us formalize it:

Theorem 4.3.1 (F5-criterion)**.** *If x^α is the leading monomial of a row of the matrix $\widetilde{M}_{d-d_i,i-1}$, then the polynomial $x^\alpha f_i$ belongs to the vector space*

$$\langle \widetilde{M}_{d,i-1} \cup \{x^\beta f_i : \deg(x^\beta f_i) = d, x^\beta < x^\alpha\} \rangle_k$$

Proof. Suppose that x^α is the leading monomial of a row of the matrix $\widetilde{M}_{d-d_i,i-1}$, then the corresponding polynomial looks like $h = ax^\alpha + g$ where $a \neq 0$ and g is a polynomial consisting of monomials that are smaller than x^α. Moreover, $h \in \langle f_1, \ldots, f_{i-1} \rangle$. It follows that hf_i is still in $\langle f_1, \ldots, f_{i-1} \rangle$. This leads to the following composition

$$x^\alpha f_i = \frac{1}{a} h f_i - \frac{1}{a} g f_i$$

where $\frac{hf_i}{a} \in \langle \widetilde{M}_{d,i-1} \rangle_k$ and $\frac{1}{a} g f_i \in \langle \{x^\beta f_i : |\beta| = d - d_i, x^\beta < x^\alpha\} \rangle_k$. □

In other words, theorem 4.3.1 is saying that the row corresponding to $x^\alpha f_i$ will reduce to zero after Gaussian elimination if the assumptions are satisfied. Clearly we do not want this to happen, so during the algorithm we test whether the assumptions are satisfied and if this is the case we reject the row. Combining all these ideas leads to the Matrix-F5 algorithm, given on the next page.

Theorem 4.3.2. *Algorithm 9 terminates in a finite number of steps and outputs a D-Gröbner basis.*

Proof. Theorem 4.1.11 says that Lazard's algorithm 7 is correct. Matrix-F5 is a modification of this algorithm, which leaves the row space invariant:

- Reusing linear dependencies: evidently this does not change the row space.
- Modified Gaussian elimination: The echelon form obtained by the modification is a permutation of an echelon form obtained by pivoting. Again, this does not change the row space.
- The F5-criterion: A row satisfying the conditions of theorem 4.3.1 is a linear combination of the previous rows. In other words, leaving it out does not change the row space.

□

input : $F = \{f_1, \ldots, f_m\}$ a sequence of polynomials, a positive integer D, and a monomial order $\leq$ on R
output: A D-Gröbner basis for $\langle F \rangle$ with respect to $\leq$
begin
  $G := [\,]$, $H := [\,]$, $T_1 :=$ all monomials of degree 1 ;
  for $d := 1$ **to** D **do**
    $M_d := [\,]$, $L_d := [\,]$;
    for $i := 1$ **to** m **do**
      if $\deg(f_i) = d$ **then**
        `Append(`M_d, f_i`)`, `Append(`$L_d, (1, i, |M_d|)$`)` ;
      else if $\deg(f_i) < d$ **then**
        for $(t, m, r) \in L_{d-1}$ **where** $m = i$ **do**
          for $x \in T_1$ **do**
            $V :=$ variables in t ;
            if $x < \max V$ **then**
              continue;
            end
            $found := false$;
            for $(t_2, m_2, r_2) \in L_{d-\deg(f_i)}$ **where** $m_2 < m$ **do**
              if $\mathrm{lm}(M_{d-\deg(f_i)}[r_2]) = xt$ **then** // F5-criterion
                $found := true$, **break**;
              end
            end
            if $\neg found$ **then**
              `Append(`$M_d, M_{d-1}[r]$`)`,`Append(`$L_d, (xt, i, |M_d|)$`)` ;
            end
          end
        end
      end
    end
    $\widetilde{M_d} :=$ `ModifiedGaussianElimination(`M_d`)` ;
    $G := G \cup \{f \in \widetilde{M_d} : \nexists g \in M_d \text{ s.t. } \mathrm{lm}(g) = \mathrm{lm}(f) \text{ and } \sigma(g) = \sigma(f)\}$;
  end
  return G
end

Algorithm 9: Matrix-F5 algorithm

4.4 A modification: the syzygy criterion

When a row reduces to zero during the execution of the Matrix-F5 algorithm, multiples of this row that are greater in the signature order are not generated as they will reduce to zero. This follows from the discussion in the section on using known linear dependences. However, more is true. We show for the first time that it is possible to translate the syzygy criterion, described in the last chapter, such that it works with Matrix-F5 (We have not seen this in any work by Faugère's group, but it has also been incorporated into an algorithm by Gao et al. [GVIW16], which we will discuss in the next chapter). We propose the following modification of the original Matrix-F5 algorithm: whenever a row reduces to zero, we detect it and store the corresponding signature in a list. This gives us useful information for predicting and thus preventing new zero reductions. The result is captured by the following theorem.

Theorem 4.4.1 (Syzygy criterion)**.** *If the row with signature $x^\alpha e_i$ reduces to zero and x^α divides x^β, then the row with signature $x^\beta e_i$ reduces to zero.*

Proof. Since the row with signature $x^\alpha e_i$ reduces to zero there exist $g_1, \ldots, g_i \in k[x_1, \ldots, x_n]$ with $\mathrm{lm}(g_i) \leq x^\alpha$ such that $x^\alpha f_i = \sum_{k=1}^{i} g_k f_k$. Moreover, there exists a monomial x^γ such that $x^\beta = x^\gamma x^\alpha$. It follows that $x^\beta f_i = \sum_{k=1}^{i} x^\gamma g_k f_k$. By properties of monomial orders we see that $x^\gamma \mathrm{lm}(g_i) \leq x^\beta$. From this we deduce that the row with signature $x^\beta e_i$ is a linear combination of rows preceding it. As a consequence this row reduces to zero. □

Theorem 4.4.2. *Algorithm 10 terminates in a finite number of steps and outputs a D-Gröbner basis.*

Proof. This follows from theorem 4.3.2 and the fact that a row satisfying the conditions of theorem 4.4.1 a a linear combination of the previous rows. In other words, leaving it out does not change the row space. □

Example 4.4.3. *Here is an example of a sequence for which the syzygy criterion prevents additional zero reductions compared to the original Matrix-F5. We demonstrate this by means of an interactive Magma session.*

```
> P<x,y,z> := PolynomialRing(Rationals(), 3, "grevlex");
> f1 := x^2*y + y^3 - y*z^2 - z^3;
> f2 := -y^3 + x*y*z;
> f3 := y^2*z^2 - z^4;
> f4 := x^6;
> MatrixF5([f1,f2,f3,f4], 8);
Row with signature  <3, y^2, 23>
reduced to zero.
Row with signature  <3, x^2, 25>
reduced to zero.
Row with signature  <3, y^2*z, 31>
reduced to zero.
Row with signature  <3, x^2*z, 33>
reduced to zero.
Row with signature  <3, y^2*z^2, 40>
reduced to zero.
```

input : $F = \{f_1, \ldots, f_m\}$ a sequence of polynomials, a positive integer D, and a monomial order $\leq$ on R
output: A D-Gröbner basis for $\langle F \rangle$ with respect to $\leq$
begin
  $G := [\,]$, $H := [\,]$, $T_1 :=$ all monomials of degree 1 ;
  for $d := 1$ **to** D **do**
    $M_d := [\,]$, $L_d := [\,]$;
    for $i := 1$ **to** m **do**
      if $\deg(f_i) = d$ **then**
        `Append`(M_d, f_i), `Append`$(L_d, (1, i, |M_d|))$;
      else if $\deg(f_i) < d$ **then**
        for $(t, m, r) \in L_{d-1}$ **where** $m = i$ **do**
          for $x \in T_1$ **do**
            $V :=$ variables in t ;
            if $x < \max V$ **then**
              continue;
            end
            $found := false$;
            for $(m, j) \in H$ **do**
              if $j = i$ *and* m *divides* xt **then** // Syzygy criterion
                $found := true$, **break**;
              end
            end
            if $\neg found$ **then**
              for $(t_2, m_2, r_2) \in L_{d-\deg(f_i)}$ **where** $m_2 < m$ **do**
                if $\mathrm{lm}(M_{d-\deg(f_i)}[r_2]) = xt$ **then** // F5-criterion
                  $found := true$, **break**;
                end
              end
            end
            if $\neg found$ **then**
              `Append`$(M_d, M_{d-1}[r])$,`Append`$(L_d, (xt, i, |M_d|))$;
            end
          end
        end
      end
    end
    $\widetilde{M_d} :=$ `ModifiedGaussianElimination`(M_d) ;
    For every row that is zero in $\widetilde{M_d}$, store its signature (m, i) in H ;
    $G := G \cup \{f \in \widetilde{M_d} : \nexists g \in M_d \text{ s.t. } \mathrm{lm}(g) = \mathrm{lm}(f) \text{ and } \sigma(g) = \sigma(f)\}$;
  end
  return G
end

Algorithm 10: Modified Matrix-F5 algorithm

```
Row with signature  <3, x^2*z^2, 42>
reduced to zero.
Row with signature  <4, y^2, 46>
reduced to zero.
[
    x^2*y + y^3 - y*z^2 - z^3,
    -y^3 + x*y*z,
    y^2*z^2 - z^4,
    x^6,
    y^2*z^3 - x*z^4,
    x*z^4 - z^5,
    x*y*z^3 - y*z^4,
    y^2*z^4 - x*z^5,
    y*z^5 - z^6,
    y^2*z^5 - x*z^6,
    x^3*z^4 + x^2*z^5 - x*z^6 - y*z^6,
    y*z^6 - z^7,
    x^4*z^3,
    y^2*z^6 - x*z^7,
    x^3*z^5 + x^2*z^6 - x*z^7 - y*z^7,
    y*z^7 - z^8,
    z^8
]
```

The original Matrix-F5 performs seven useless reductions, as the output shows. Let's see what happens when we incorporate the syzygy criterion.

```
> ModifiedMatrixF5([f1,f2,f3,f4], 8);
Row with signature  <3, y^2, 23>
reduced to zero.
Row with signature  <3, x^2, 25>
reduced to zero.
Row with signature  <4, y^2, 44>
reduced to zero.
[
    x^2*y + y^3 - y*z^2 - z^3,
    -y^3 + x*y*z,
    y^2*z^2 - z^4,
    x^6,
    y^2*z^3 - x*z^4,
    x*z^4 - z^5,
    x*y*z^3 - y*z^4,
    y^2*z^4 - x*z^5,
    y*z^5 - z^6,
    y^2*z^5 - x*z^6,
    x^3*z^4 + x^2*z^5 - x*z^6 - y*z^6,
    y*z^6 - z^7,
    x^4*z^3,
```

```
    y^2*z^6 - x*z^7,
    x^3*z^5 + x^2*z^6 - x*z^7 - y*z^7,
    y*z^7 - z^8,
    z^8
]
```

Only three useless reductions have been performed this time!

4.5 Regular sequences in the context of Matrix-F5

The content of this section is that regular sequences are nice in the context of Matrix-F5. We show that if the input sequence is regular, then no zero reductions take place.

Proposition 4.5.1. *Suppose that $\langle f_1, \ldots, f_m \rangle$ are homogeneous polynomials in A_n. For large enough D, the syzygy module of $\langle f_1, \ldots, f_m \rangle$ is the union of the module of principal syzygies and the syzygies which correspond to reductions to zero during the execution of the algorithm Matrix-F5.*

Proof. Let $(g_1, \ldots, g_m)$ be a non-principal syzygy. We will show that it gives rise to a reduction to zero. First, observe that Matrix-F5 doesn't detect this syzygy as it is non-trivial. Since it is a syzygy we have that $\sum_{i=1}^{m} g_i f_i = 0$. Let j be the largest index such that g_j is nonzero. Let $d = \deg(g_j f_j)$ and $t = \mathrm{lt}(g_j)$. In the matrix $M_{d,m}$ the row corresponding to tf_j is reduced to zero by a suitable combination of the rows above it. To prove the converse, we first remark that the principal syzygies are trivially in the module of syzygies. Next, suppose that the row tf_j in the matrix $M_{d,m}$ reduces to zero. We will show that it corresponds to a non-trivial syzygy. Since tf_j reduces to zero, there exist g_i with $i \leq j$ such that $tf_j - \sum_{i=1}^{j} g_i f_i = 0$. Hence we have the relation $\sum_{i=1}^{j-1} g_i f_i + (g_j - t) f_j = 0$ which shows that $(g_1, \ldots, g_{j-1}, g_j - t, 0, \ldots, 0)$ is a syzygy on $f_1, \ldots, f_m$. □

Corollary 4.5.2. *If the sequence of input polynomials is a regular sequence, then no reduction to zero takes place during execution of the Matrix-F5 algorithm.*

Proof. If the input sequence is a regular sequence, then theorem 2.7.16 says that every syzygy is trivial. Hence proposition 4.5.1 implies that no reductions to zero occur. □

Example 4.5.3. *The implementation of Matrix-F5 in Magma given in the appendix prints the signature of every row that is reduced to zero. By means of an interactive Magma session we will now give an example of Matrix-F5's output on a regular sequence.*

```
> P<x,y,z,h> := PolynomialRing(Rationals(), 4, "grevlex");
> f1 := x^2 + y^2 - 2*x*z - 2*y*z + z^2 + h^2;
> f2 := x^2+x*y+y*z-z^2-2*h^2;
> f3 := x^2-y^2+2*y*z-2*z^2;
>
> MatrixF5([f1,f2,f3], 4);
[
    x^2 + y^2 - 2*x*z - 2*y*z + z^2 + h^2,
    x^2 + x*y + y*z - z^2 - 2*h^2,
    x^2 - y^2 + 2*y*z - 2*z^2,
    x*y - y^2 + 2*x*z + 3*y*z - 2*z^2 - 3*h^2,
    y^2 - x*z - 2*y*z + 3/2*z^2 + 1/2*h^2,
    y^3 - 5*y^2*z + 6*x*z^2 + 10*y*z^2 - 6*z^3 + 3/2*x*h^2 + 2*y*h^2 - 19/2*z*h^2,
    x*z^2 + 3/4*y*z^2 - 1/2*z^3 + 3/4*x*h^2 + 3/4*y*h^2 - 11/4*z*h^2,
    y*z^2 - 2*z^3 + 11/3*x*h^2 - 5/3*y*h^2 - z*h^2,
    y^3*h - 5*y^2*z*h + 6*x*z^2*h + 10*y*z^2*h - 6*z^3*h + 3/2*x*h^3 + 2*y*h^3
- 19/2*z*h^3,
```

```
    y^3*z - 5*y^2*z^2 + 6*x*z^3 + 10*y*z^3 - 6*z^4 + 3/2*x*z*h^2 + 2*y*z*h^2 -
19/2*z^2*h^2,
    y^4 - 9*y^2*z^2 + 18*x*z^3 + 26*y*z^3 - 18*z^4 + 7/2*y^2*h^2 + 9/2*x*z*h^2
- 4*y*z*h^2 - 53/2*z^2*h^2 +
        9/2*h^4,
    x*z^2*h + 3/4*y*z^2*h - 1/2*z^3*h + 3/4*x*h^3 + 3/4*y*h^3 - 11/4*z*h^3,
    y*z^2*h - 2*z^3*h + 11/3*x*h^3 - 5/3*y*h^3 - z*h^3,
    x*z^3 + 3/4*y*z^3 - 1/2*z^4 + 3/4*x*z*h^2 + 3/4*y*z*h^2 - 11/4*z^2*h^2,
    y*z^3 - 2*z^4 + 11/3*x*z*h^2 - 5/3*y*z*h^2 - z^2*h^2,
    z^4 + 4/3*x*z*h^2 + 4*y*z*h^2 - 7/3*z^2*h^2 - 4*h^4
]
```

Indeed, no reductions to zero have taken place.

4.6 Semi-regular sequences: a generalization of regular sequences

As we remarked in the previous section, a system of homogeneous polynomials having more equations than variables can never be regular. In this section we extend the notion of a regular sequence to that of a semi-regular sequence. The intuition behind this notion is that a sequence is semi-regular if it is regular up to a certain degree. This degree is characterized by the first occurrence of a non-trivial relation between the polynomials constituting this sequence.

Definition 4.6.1 (Homogeneous semi-regular sequence)**.** *A sequence $f_1, \ldots, f_m$ of homogeneous polynomials in A_n is called semi-regular if it satisfies the following conditions:*

1. $\langle f_1, \ldots, f_m \rangle \neq A_n$.
2. *If $gf_i = 0$ in $A_n/\langle f_1, \ldots, f_{i-1} \rangle$ and $\deg(gf_i) < d_{reg}(I)$, then $g = 0$ in $A_n/\langle f_1, \ldots, f_{i-1} \rangle$ for all $g \in A_n$ and $1 \leq i \leq m$.*

We can extend this notion to the inhomogeneous case.

Definition 4.6.2 (Affine semi-regular sequence)**.** *Writing f_i^h for the homogeneous part of f_i of largest degree, we call a sequence $f_1, \ldots, f_m$ semi-regular if the corresponding homogeneous sequence $f_1^h, \ldots, f_m^h$ is semi-regular. The degree of regularity of $\langle f_1, \ldots, f_m \rangle$ is given by the degree of regularity of $\langle f_1^h, \ldots, f_m^h \rangle$.*

Given a power series $\sum_{n=0}^{\infty} a_n t^n$ we define $[\sum_{n=0}^{\infty} a_n t^n] = \sum_{n=0}^{\infty} b_n t^n$ where $b_n = a_n$ if $a_k > 0$ for all $0 \leq k \leq n$, and $b_n = 0$ otherwise.

Theorem 4.6.3. *Let $f_1, \ldots, f_m$ be a sequence of polynomials in A_n. Then*

1. *For $m \leq n$ the notions of semi-regularity and regularity coincide.*
2. *If $f_1, \ldots, f_m$ is a semi-regular sequence, then its Hilbert series is equal to $H_{A/I}(t) = \left[\frac{\prod_{i=1}^{m}(1-t^{\deg(f_i)})}{(1-t)^n}\right]$.*
3. *The index of regularity of the ideal defined by a semi-regular sequence is equal to the index of the first non-positive coefficient in $\frac{\prod_{i=1}^{m}(1-t^{\deg(f_i)})}{(1-t)^n}$.*

4. *If $f_1, \ldots, f_m$ is a semi-regular sequence, then there is no reduction to 0 during the execution of the Matrix-F5 algorithm for degrees smaller than $d_{reg}(I)$.*

Proof. See [BFSY05] proposition 5. □

Unlike regular sequences, semi-regular sequences need not remain semi-regular after a permutation of the indices.

4.7 The inhomogeneous case

In practice, many systems of equations are comprised of inhomogeneous polynomials. We would still like to be able to run the Matrix-F5 algorithm on such input in order to compute a Gröbner basis.

4.7.1 Homogenization

Suppose now that we want to compute a Gröbner basis for $I = \langle f_1, \ldots, f_m \rangle$. Our strategy will be to homogenize the generators of I and then compute a Gröbner basis for the ideal generated by the resulting polynomials. It turns out that if we dehomogenize the elements of this Gröbner basis we end up with a Gröbner basis for I. First, we need some terminology.

Definition 4.7.1 (Good extension of an ordering)**.** *Let $>$ be a monomial order on $A = k[x_1, \ldots, x_n]$ and let $>_1$ be a monomial order on $A[y]$. We call $>_1$ a good extension of $>$ if $\mathrm{lm}_>(f) = \mathrm{lm}_{>_1}(h(f))$ for all $f \in A$.*

Observe that a monomial order in which the homogenization variable is smaller than all the other variables is a good extension. We are now ready to state the theorem.

Proposition 4.7.2. *Let $I = \langle f_1, \ldots, f_m \rangle$ be an ideal in $A = k[x_1, \ldots, x_n]$ and fix a monomial order $>$. Let $>_1$ be a good extension of $>$ to $A[y]$. If $G = \{g_1, \ldots, g_t\}$ is a Gröbner basis for $\langle h(f_1), \ldots, h(f_m) \rangle$ with respect to $>_1$, then $a(G) = \{a(g_1), \ldots, a(g_t)\}$ is a Gröbner basis for I with respect to $>$.*

Proof. Let $f \in I$, then $h(f) \in \langle h(f_1), \ldots, h(f_m) \rangle$. Since G is a Gröbner basis for $\langle h(f_1), \ldots, h(f_m) \rangle$ it follows that there exists a $g \in G$ such that $\mathrm{lm}(g)$ divides $\mathrm{lm}(h(f))$. But $>_1$ is a good extension of $>$, so $\mathrm{lm}(h(f)) = \mathrm{lm}(f) \in A$. It follows that $\mathrm{lm}(g) \in A$ and therefore $\mathrm{lm}(g) = \mathrm{lm}(a(g))$. Thus $\mathrm{lm}(a(g))$ divides $\mathrm{lm}(f)$. □

A problem with this approach is that it introduces new solutions, so-called solutions at infinity, which are not present in the original system. Hence, a system of equations having a single unique solution may actually be much harder to solve after homogenization. This is something we want to avoid if possible. Fortunately, we can avoid it by considering the sugar degree [GMN+91].

4.7.2 Sugar degree

Definition 4.7.3 (Sugar degree)**.** *Let $f_1, \ldots, f_m \in A_n$ be the input to a Gröbner basis algorithm, and let f be a polynomial appearing during the execution of the algorithm. Then f takes on any of the following three forms, and we define the sugar degree accordingly:*

- *If* $f = f_i$, *then* s-deg$(f) :=$ deg(f_i) *for* $i = 1, \ldots, m$.
- *If* $f = x^\alpha g$, *then* s-deg$(f) := |\alpha| +$ s-deg(g).
- *If* $f = g + h$, *then* s-deg$(f) :=$ max$\{$s-deg(g), s-deg$(h)\}$.

It is a kind of "phantom" degree in that it mimics the degree the polynomial would have had it we had started by homogenizing the input sequence first. The popularity of the sugar degree stems from the fact that it is easy to implement, and with it we can avoid the overhead of homogenizing. Next, we explore the notion of a signature-degree, as there turns out to be a connection between the notions of degree.

Definition 4.7.4. *Let* $F = \{f_1, \ldots, f_m\}$ *be a sequence of polynomials in* A_n *and let* $p = (x^\alpha e_i, f) \in R^m \times \langle F \rangle$ *be a labeled polynomial. We define the signature-degree of* p *as*

$$\text{sig-deg}(p) = |\alpha| + \deg(f_i)$$

Indeed, they are equal to each other:

Proposition 4.7.5. *Let* $p = (\sigma(f), f)$ *be a labeled polynomial appearing during the computation of a signature-based Gröbner basis. Then* sig-deg$(p) =$ s-deg(f).

Proof. See [Ede13] theorem 4.2. □

A consequence from this proposition is that by default Matrix-F5 9 processes polynomials by increasing sugar degree. From an algorithmic point of view this is good news. We won't have the computational overhead of homogenizing the input and substituting sugar-degree for degree in the proof of the algorithm shows that it correctly computes a Gröbner basis.

4.7.3 Degree fall

Now, if we do not want to homogenize the input, nor use the sugar degree, then the situation changes a little bit.

Definition 4.7.6 (Macaulay matrix in the inhomogeneous case)**.** *The Macaulay matrix in degree* d *of the sequence of inhomogeneous polynomials* $f_1, \ldots, f_m \in A_n$ *is defined as before, except that its columns are indexed by all monomials of degree at most* d.

When the algorithm computes an echelon form of the Macaulay matrix in degree d it might be the case that one of the rows in this echelon form corresponds to a polynomial having degree $d' < d$. We call this behavior degree fall.

Definition 4.7.7 (Degree fall)**.** *We say that a degree fall occurs during the execution of Matrix-F5 9 if the row corresponding to a polynomial of degree* d *is reduced to a row corresponding to a polynomial of degree* $d' < d$.

Example 4.7.8. *If* $f = xy + y$ *and* $g = x$, *then* $y = f - yg$ *will be computed when the algorithm reaches degree 2, even though* y *has degree 1.*

As the example illustrates, in the precense of a degree fall it might no longer be the case that an echelon form of $M_{d,i}$ contains a d-Gröbner basis for $\langle f_1, \ldots, f_m \rangle$. An immediate consequence is that the F5-criterion no longer holds. There's a subtlety in this statement: of course the criterion might still predict some zero reductions (perhaps even all of them), but if not all polynomials of degree d have been processed than this is not a likely scenario.

Remark 15. Observe that a degree fall takes place if the homogeneous part of highest degree of some polynomial is reduced to zero.

The remark above immediately leads to the following proposition:

Proposition 4.7.9. *If $f_1^h, \ldots, f_m^h$ is a regular sequence (recall that f_i^h denotes the homogeneous part of largest degree of f_i), then no degree fall occurs.*

What is true, however, is that for large enough d the algorithm still outputs a Gröbner basis. This is a good news, since it means that the algorithm doesn't care whether the input is homogeneous or not. This doesn't mean that it's sensible to ignore degree falls, as they can have a major impact on the performance of the algorithm. Instead, we can detect when a degree fall takes place and inject the corresponding row into the matrix of the relevant degree and perform suitable reductions on it.
When computing an echelon form for $M_{d,m}$ we can add to any row corresponding to some f_i whose leading term has degree $d' < d$ a linear combination of rows from $M_{d',i}$. Doing this will greatly speed up the algorithm.
It should be noted that in the case of degree fall it is better to use an algorithm such as Faugère's F5 [Fau99] which always reduces S-polynomials of the lowest degree first.

4.8 Complexity

The reductions take place implicitly when computing a echelon form. It is not surprising that the running time of the Matrix-F5 algorithm is dominated by the cost of performing matrix operations on the largest matrix appearing during the execution of the algorithm. We have the following bound, but it is not very sharp since it does not take into account the syzygy- and F5-criterion.

Proposition 4.8.1. *Let $I = \langle f_1, \ldots, f_m \rangle \subseteq k[x_1, \ldots, x_n]$ be an ideal generated by homogeneous polynomials of degrees $d_1, \ldots, d_m$ respectively. We can compute a D-Gröbner basis for I using Matrix-F5 within*

$$\mathcal{O}\left(m\binom{n+D}{D}^{\omega}\right)$$

field operations (addition, subtraction, multiplication, division). Here ω is a parameter such that two $n \times n$ matrices over k can be multiplied together in $\mathcal{O}(n^\omega)$ field operations (e.g., using the well known Strassen algorithm [Str69] we have $\omega = \log_2(7)$. The fastest algorithm to date is by Le Gall [Gal12]).

Proof. The Gaussian elimination algorithm computes an echelon form of an $l \times c$ matrix of rank r with coefficients in k within $\mathcal{O}(lcr^{\omega-2})$ field operations (see [Sto00] proposition 2.19). This can be upper bounded by $\mathcal{O}(lc^\omega - 1)$ since $r \leq c$. Now, the Macaulay matrix of $f_1, \ldots, f_m$ in degree d has $\binom{n+d-1}{n-1}$ columns and $\sum_{i=1}^{m}\left(\binom{n+d-\deg(f_i)-1}{n-1}\right)$ rows. Since we are computing

the echelon form of such Macaulay matrices up to and including degree D we get the following:

$$\begin{aligned}\sum_{d=0}^{D}\left[\left(\sum_{i=1}^{m}\binom{n+d-\deg(f_i)-1}{n-1}\right)\binom{n+d-1}{n-1}^{\omega-1}\right] &\leq m\sum_{d=0}^{D}\binom{n+d-1}{n-1}^{\omega}\\ &\leq m\left(\sum_{d=0}^{D}\binom{n+d-1}{n-1}\right)^{\omega} \text{ since } \omega>1\\ &\leq m\left((D+1)\binom{n+D-1}{n-1}\right)^{\omega}\\ &\leq m\binom{n+D}{n}^{\omega}\end{aligned}$$

and the result follows. □

Corollary 4.8.2. *Let $I=\langle f_1,\ldots,f_m\rangle$ be an ideal in A_n. We can compute a Gröbner basis for I using Matrix-F5 within*

$$\mathcal{O}\left(m\binom{n+d_{reg}(I)}{d_{reg}(I)}^{\omega}\right)$$

field operations.

Proof. This follows immediately from proposition 4.1.12. □

In the inhomogeneous case we have to deal with degree falls, making it difficult to give a precise bound on the number of field operations. However, if the sequence $f_1^h,\ldots,f_m^h$ is regular, then the bound remains to hold with $d_{reg}(I)$ replaced by $d_{reg}(J)$ where J is the ideal generated by $f_1^h,\ldots,f_m^h$.

4.9 Choosing D

We are interested in values for D such that Matrix-F5 outputs a Gröbner basis. In the literature, one often finds a result by Dubé [Dub90], stating that

$$D=2\left(\frac{d^2}{2}+d\right)^{2^{n-2}}$$

with $d=\max\{\deg(f_i):1\leq i\leq m\}$ is sufficient. This bound is very loose.

4.10 An improvement for sequences over $\mathbb{F}_2$

In the case that the field is $\mathbb{F}_2$ we can consider the ring $\mathbb{F}_2[x_1,\ldots,x_n]/\langle x_1^2,\ldots,x_n^2\rangle$ of squarefree polynomials in $x_1,\ldots,x_n$. When algorithm Matrix-F5 9 constructs the Macaulay matrix in degree d it multiplies the polynomials corresponding to rows appearing in the Macaulay matrix of degree $d-1$ by variables at least as large as the largest variables appearing in the signature. An immediate consequence is that this introduces squared factors. In the ring above these will reduce to zero, so they are useless. We therefore replace the line $x<\max V$ by $x\leq\max V$.

Now, any ideal in this ring is in a one-to-one correspondence with the ideals in $\mathbb{F}_2[x_1, \ldots, x_n]$ containing $\langle x_1^2, \ldots, x_n^2 \rangle$. So if we are interested in a Gröbner basis for

$$I \subseteq \mathbb{F}_2[x_1, \ldots, x_n]/\langle x_1^2, \ldots, x_n^2 \rangle,$$

then it suffices to compute a Gröbner basis for $\langle x_1^2, \ldots, x_n^2 \rangle + I \subseteq \mathbb{F}_2[x_1, \ldots, x_n]$.

Remark 16. The ideal remains the same when we permutate the generators, but since Matrix-F5 only allows signature-safe reductions (reductions by rows appearing above the row to be reduced) it is important that the relations $x_1^2, \ldots, x_n^2$ appear at the beginning of the input sequence.

The remark above shows that during the execution of the Matrix-F5 algorithm rows will be reduced to zero as a result of the relations $x_i^2 = 0, \quad 1 \leq i \leq n$. In her thesis, Bardet [Bar04] describes a criterion to prevent such reductions. However, she does not give a proof. We have tried to fill this gap.

Theorem 4.10.1 (Frobenius criterion)**.** *Let t be the leading term of a polynomial corresponding to a row in the matrix $\widetilde{M}_{d-d_i,i}$ with label (s, i), then the row with label (t, i) in the matrix $M_{d,i}$ is a linear combination of rows preceding it.*

Proof. Let $t = \mathrm{lm}(h)$ with $h = \sum_{k=1}^{i} h_k f_k$. Observe that we can eliminate any squares from a polynomial by subtracting a suitable combination of the first n rows from the row associated with it. We thus have the following decomposition,

$$tf_i = \sum_{k=1}^{i-1} f_i h_k f_k + h_i f_i^2 + (t-h) f_i = \sum_{k=1}^{i-1} f_i \widetilde{h_k} f_k + (t-h) f_i$$

where the first term belongs to $M_{d,i-1}$ and the second term is a linear combination of rows with smaller signature, as $\mathrm{lm}(t-h) < t$. □

According to Bardet the criterion remains to hold in the case that we add the homogenized version of the field equations $x_i^2 = x_i x_{n+1}$ (so $x_i > x_{n+1}$ for all $1 \leq i \leq n$) instead of $x_i^2 = 0$, but we were unable to prove this in the general case. We will give a proof in the case that the monomial order is the degree reverse lexicographic order. Hence the setting is the ring $\mathbb{F}_2[x_1, \ldots, x_n, x_{n+1}]/\langle x_1^2 - x_1 x_{n+1}, \ldots, x_n^2 - x_n x_{n+1} \rangle$.

Theorem 4.10.2 (Frobenius criterion (second version))**.** *Let t be the leading term of a polynomial corresponding to a row in the matrix $\widetilde{M}_{d-d_i,i}$ with label (s, i), then the row with label (t, i) in the matrix $M_{d,i}$ is a linear combination of rows preceding it.*

Proof. Let $t = \mathrm{lm}(h)$ with $h = \sum_{k=1}^{i} h_k f_k$. Observe that we can eliminate any squares from a polynomial by subtracting a suitable combination of the first n rows from the row associated with it. We thus have the following decomposition,

$$tf_i = \sum_{k=1}^{i-1} f_i h_k f_k + h_i f_i^2 + (t-h) f_i = \sum_{k=1}^{i-1} f_i \widetilde{h_k} f_k + h_i x_{n+1}^{\deg(f_i)} f_i + (t-h) f_i$$

where the first term belongs to $M_{d,i-1}$ and the second term is a linear combination of rows with smaller signature, as $\mathrm{lm}(t-h) < t$, as before. However, we have another term, $h_i x_{n+1}^{\deg(f_i)} f_i$.

Without loss of generality, we may assume that $\min\{\deg(f_i) : 1 \leq i \leq m\} \leq 2$. This follows from the observation that the polynomials correspond to a system of equations, so if any of the f_i have degree equal to 1, then it's possible to eliminate a number of variables and substitute them in the equations of higher degree. Hence $\deg(x_{n+1}^{\deg(f_i)}) \geq 2$. But we know that x_{n+1} appears at most once in t, by design. Hence $\mathrm{lm}(h_i x_{n+1}^{\deg(f_i)}) < t$, and it follows that t is a linear combination of rows preceding it. □

4.11 An improvement for sequences of bilinear forms

The primary focus of current research in the field of Gröbner basis algorithms is to make use of the algebraic structure of your input polynomials in order to predict more zero reductions or speed up the algorithm by other means. In [FSEDS11] the authors focus on bihomogeneous systems and are able to generate more syzygies in a preprocessing step which are then detected by a new criterion. First we need some new terminology.

Definition 4.11.1 (Bihomogeneous polynomial)**.** *A polynomial f in $k[x_1, \ldots, x_n, y_1, \ldots, y_m]$ is said to be bihomogeneous of bidegree (d, e) if it has the following shape*

$$f(x_1, \ldots, x_n, y_1, \ldots, y_m) = \sum_{\substack{\alpha \in \mathbb{N}^n, \beta \in \mathbb{N}^m \\ |\alpha| = d \\ |\beta| = e}} a_{\alpha,\beta} x^\alpha y^\beta.$$

Proposition 4.11.2. *If f is bihomogeneous of bidegree (d, e), then*

$$f(\lambda x_1, \ldots, \lambda x_n, \mu y_1, \ldots, \mu y_m) = \lambda^d \mu^e f(x_1, \ldots, x_n, y_1, \ldots, y_m).$$

If k is algebraically closed, then the reverse direction holds as well.

We will focus on polynomials of bidegree $(1, 1)$, i.e., bilinear polynomials. We will follow the exposition in [FSEDS11] and focus on the general overview as opposed to the details, as we are mainly interested in the extension that is given to Matrix-F5 9.
Now, let $F = \{f_1, f_2, f_3, f_4\} \subseteq \mathbb{Q}[x_1, x_2, x_3, y_1, y_2, y_3]$ be four bilinear polynomials and put $I = \langle F \rangle$. Its algebraic set is denoted $V(I) \subseteq \mathbb{C}^6$. Finally, let $>$ be the degree reverse lexicographic order with $x_1 > x_2 > x_3 > y_1 > y_2 > y_3$.
Since f_1, f_2, f_3, f_4 are bilinear we have that

$$f_i(a_1, a_2, a_3, 0, 0, 0) = 0 \cdot f_i(a_1, a_2, a_3, 0, 0, 0) = 0$$

for all $(a_1, a_2, a_3) \in \mathbb{C}^3$ and for all $1 \leq i \leq 4$. It follows that $V(I)$ contains the subspace of dimension 3 defined by $y_1 = y_2 = y_3 = 0$. Hence the dimension of $V(I)$ is at least 3. If f_1, f_2, f_3, f_4 were regular, then theorem 2.7.17 combined with the Nullstellensatz says that $\dim V(I) = \dim I(V(I)) = \dim \sqrt{I} = \dim I = 2$, but we have just showed that it is at least 3. This shows that f_1, f_2, f_3, f_4 is not a regular sequence. Consequently, there are reductions to zero which are not prevented by the F5-criterion during the execution of the Matrix-F5 algorithm 9.
Consider the following submatrices of the Jacobian of the function defined by F.

$$Jac_x(F) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} \end{pmatrix} \text{ and } Jac_y(F) = \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \frac{\partial f_1}{\partial y_3} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \frac{\partial f_2}{\partial y_3} \\ \frac{\partial f_3}{\partial y_1} & \frac{\partial f_3}{\partial y_2} & \frac{\partial f_3}{\partial y_3} \\ \frac{\partial f_4}{\partial y_1} & \frac{\partial f_4}{\partial y_2} & \frac{\partial f_4}{\partial y_3} \end{pmatrix}$$

Write $x = (x_1, x_2, x_3)$ and $y = (y_1, y_2, y_3)$. Moreover, let $q = (q_1, q_2, q_3, q_4) \in \mathbb{Q}[x_1, x_2, x_3, y_1, y_2, y_3]$. Then we have the following equalities

$$q \cdot Jac_x(F) \cdot x = \sum_{i=1}^{4} q_i f_i \text{ and } q \cdot Jac_y(F) \cdot y = \sum_{i=1}^{4} q_i f_i$$

which follow immediately from the bilinearity of the f_i.
Now, suppose that we enlarge the matrix $Jac_x(F)$ by adding the vector $(0, 0, 0, 0)^T$ at the beginning. The resulting 4×4 matrix, which we denote by J, is clearly singular. Moreover, as a consequence of the cofactor expansion of the determinant we have the following equality.

$$\text{adj}(J)J = \det(J)I = 0$$

Therefore, every row of $\text{adj}(J)$ is in the left nullspace of J. In particular, letting M_{ij} denote the (i, j) minor of J, we have that

$$q = (M_{11}, -M_{21}, M_{31}, -M_{41})$$

is in this left nullspace. It follows that q is also in the left nullspace of $Jac_x(F)$. But then we have

$$0 = q \cdot Jac_x(F) \cdot x = \sum_{i=1}^{4} q_i f_i.$$

In other words, v is a syzygy on f_1, f_2, f_3, f_4. Equivalently, the row with label $(\text{lm}(M_{41}), f_4)$ will reduce to zero during the execution of the Matrix-F5 algorithm. By symmetry, this example also holds for the case that the matrix is $Jac_y(F)$.
The authors continue to generalize this example. We will show a number of their results and end with a practical criterion that can be used to extend the Matrix-F5 algorithm in the case that the input sequence is comprised of bilinear polynomials. All of the results below also hold when we replace x by y and n by m, because of symmetry.

Theorem 4.11.3. *Let $i > n$ and let h be a linear combination of maximal minors of* JacobianMatrix$(x, [f_1, \ldots, f_{i-1}])$*, then $h \in \langle f_1, \ldots, f_{i-1} \rangle : f_i$.*

Proof. See [FSEDS11] theorem 2. □

An immediate consequence of theorem 4.11.3 is that if h is a linear combination of maximal minors, then the row with label $(\text{lm}(h), f_i)$ will be reduced to zero during the execution of the Matrix-F5 algorithm (Indeed, it is a linear combination of rows prededing it). One naturally considers the ideal of all maximal minors of JacobianMatrix$(x, [f_1, \ldots, f_{i-1}])$, denoted M_x. This is a subset of the colon ideal $\langle f_1, \ldots, f_{i-1} \rangle : f_i$. This is corollary 1 in [FSEDS11]. If we want to prevent zero reductions stemming from these maximal minors, then it becomes necessary to test for membership in M_x. In other words, we need to be able to compute a Gröbner basis for M_x. This might seem contradictory, but it turns out that in the generic case it is not difficult to compute this Gröbner basis since it is a linear combination of the generators. This is captured by the following theorem.

Theorem 4.11.4. *In the general case, a Gröbner basis for M_x with respect to the degree reverse lexicographic order is obtained by computing an echelon form of the Macaulay matrix in degree n associated with the maximal minors of* JacobianMatrix$(x, [f_1, \ldots, f_{i-1}])$.

input : $(f_1, \ldots, f_m) \in k[x_1, \ldots, x_n]$.
output: A linear basis of the k-vector space $\langle f_1, \ldots, f_m \rangle$.
begin
  $M :=$ `MacaulayMatrix` $([f_1, \ldots, f_m])$;
  $M := \widetilde{M}$;
  return `Rows`(M)
end

Algorithm 11: Reduce

input : $(f_1, \ldots, f_s) \in k[x_1, \ldots, x_n, y_1, \ldots, y_m]$ such that $s \leq (n-1) + (m-1)$.
output: A sequence S of pairs (h, f_i) with $h \in \langle f_1, \ldots, f_{i-1} \rangle : f_i$ and $h \notin \langle f_1, \ldots, f_{i-1} \rangle$.
begin
  $S := [\,]$;
  for $i := 1$ **to** s **do**
    if $i > m$ **then**
      $T :=$ `Reduce(Minors(JacobianMatrix(`$y, [f_1, \ldots, f_{i-1}]$`), `$i-1$`))` ;
      for $h \in T$ **do**
        `Append(`S`, `(h, f_i)`)` ;
      end
    end
    if $i > n$ **then**
      $T :=$ `Reduce(Minors(JacobianMatrix(`$x, [f_1, \ldots, f_{i-1}]$`), `$i-1$`))` ;
      for $h \in T$ **do**
        `Append(`S`, `(h, f_i)`)` ;
      end
    end
  end
  return S
end

Algorithm 12: Preprocess

Proof. See [FSEDS11] theorem 3. □

Combining theorems 4.11.3 and 4.11.4 immediately leads to algorithm 12.
We end up with a list of pairs in the colon ideal. The content of the following theorem is that the rows associated with these pairs will reduce to zero.

Theorem 4.11.5 (Extended F5-criterion)**.** *Let $f_1, \ldots, f_s \in k[x_1, \ldots, x_n, y_1, \ldots, y_m]$ be bi-homogeneous polynomials in of bidegree $(1, 1)$ and fix a monomial ordering. Let (t, f_i) be a labeled polynomial encountered during an execution of the Matrix-F5 algorithm. Finally, let S be the output of the preprocessing algorithm. If there exists a tuple $(h, f_i) \in S$ such that $\mathrm{lm}(h) = t$, then the row with signature (t, f_i) will be reduced to zero during Gaussian elimination.*

Proof. We give the proof ourselves. By assumption, $h \in \langle f_1, \ldots, f_{i-1} \rangle : f_i$. Then there exist $g_1, \ldots, g_{i-1} \in k[x_1, \ldots, x_n, y_1, \ldots, y_m]$ such that

$$hf_i = \sum_{k=1}^{i-1} g_k f_k$$

It follows that

$$tf_i = (t - h)f_i + \sum_{k=1}^{i-1} g_k f_k$$

Since $\mathrm{lm}(t - h) < t$ it follows that the row associated with tf_i is a linear combination of rows preceding it, hence it will be reduced to zero during Gaussian elimination. □

The criterion is easily implemented, as the following piece of pseudocode demonstrates. Recall

input : A label (t, f_i) and a matrix M in row echelon form
output: A boolean value indicating whether the row with label (t, f_i) will reduce to zero
begin
  return *There is a row in M associated with a polynomial h such that $t = \mathrm{lm}(h)$ or there is $(h, f_i) \in S$ such that $t = \mathrm{lm}(h)$.*
end

Algorithm 13: Extended F5-criterion

that if $f_1, \ldots, f_m$ is a regular sequence no reduction to zero takes place during the execution of the Matrix-F5 algorithm. We want to describe the sequences for which no reduction to zero takes place in the extended version of the Matrix-F5 algorithm which we have just sketched.

Definition 4.11.6 (Biregular sequence)**.** *Let $f_1, \ldots, f_s$ be bilinear polynomials in*

$$k[x_1, \ldots, x_n, y_1, \ldots, y_m]$$

and fix a monomial order $>$ such that its restriction to $k[x_1, \ldots, x_n]$ ($k[y_1, \ldots, y_m]$) is the degree reverse lexicographic order. Moreover, we will make the assumption that $s < (n-1) + (m-1)$. We call $f_1, \ldots, f_s$ a biregular sequence if it satisfies the following recursive definition:

- *$s = 1$ or,*

- $f_1, \ldots, f_{s-1}$ *is a biregular sequence and*

$$\mathrm{lm}(\langle f_1, \ldots, f_{s-1}\rangle : f_s) = \langle Mon^x_{s-m-1}(m)\rangle + \langle Mon^y_{s-n-1}(n)\rangle + \mathrm{lm}(\langle f_1, \ldots, f_{s-1}\rangle)$$

In the above $Mon^x_n(d)$ ($Mon^y_n(d)$) *denotes the set of all monomials of degree* d *in* $k[x_1, \ldots, x_n]$ ($k[y_1, \ldots, y_n]$).

Indeed, the class of sequences which we have just described is exactly the class of biregular sequence. This is captured by the following theorem.

Theorem 4.11.7. *Let* $f_1, \ldots, f_s$ *be a biregular sequence in* $k[x_1, \ldots, x_n, y_1, \ldots, y_m]$*, then no reduction to zero takes place during the execution of the Matrix-F5 algorithm using the extended F5-criterion.*

Proof. See [FSEDS11] theorem 4. □

Example 4.11.8. *The following is an example of a biregular sequence. As the interactive Magma session shows, the algorithm never prints any reduction to zero. We stopped at* $D = 6$*, since the output forms a Gröbner basis for the ideal generated by the input sequence.*

```
> R<x0,x1,x2,y0,y1,y2,y3> := PolynomialRing(GF(7), 7, "grevlex");
> h1 := x0*y0 + 5*x1*y0 + 4*x2*y0 + 5*x0*y1 + 3*x1*y1 + x0*y2 + 4*x1*y2 + 5*x2*y2
+ 5*x0*y3 + x1*y3 + 2*x2*y3;
> h2 := 2*x0*y0 + 4*x1*y0 + 6*x2*y0 + 2*x0*y1 + 5*x1*y1 + 6*x0*y2 + 4*x2*y2 + 3*x0*y3
+ 2*x1*y3 + 4*x2*y3;
> h3 := 5*x0*y0 + 5*x1*y0 + 2*x2*y0 + 4*x0*y1 + 6*x1*y1 + 4*x2*y1 + 6*x1*y2 + 4*x2*y2
+ x0*y3 + x1*y3 + 5*x2*y3;
> h4 := 6*x0*y0 + 5*x2*y0 + 4*x0*y1 + 5*x1*y1 +  x2*y1 + x0*y2 + x1*y2 + 6*x2*y2
+ 2*x0*y3 + 4*x1*y3 + 5*x2*y3;
> h5 := 6*x0*y0 + 3*x1*y0 + 6*x2*y0 + 3*x0*y1 + 5*x2*y1 + 2*x0*y2 + 4*x1*y2 + 5*x2*y2
+ 2*x0*y3 + 4*x1*y3 + 5*x2*y3;
>
> MatrixF5Bilinear([h1,h2,h3,h4,h5], 6, 3);
[
    x0*y0 + 5*x1*y0 + 4*x2*y0 + 5*x0*y1 + 3*x1*y1 + x0*y2 + 4*x1*y2 + 5*x2*y2 +
5*x0*y3 + x1*y3 + 2*x2*y3,
    2*x0*y0 + 4*x1*y0 + 6*x2*y0 + 2*x0*y1 + 5*x1*y1 + 6*x0*y2 + 4*x2*y2 + 3*x0*y3
+ 2*x1*y3 + 4*x2*y3,
    5*x0*y0 + 5*x1*y0 + 2*x2*y0 + 4*x0*y1 + 6*x1*y1 + 4*x2*y1 + 6*x1*y2 + 4*x2*y2
+ x0*y3 + x1*y3 + 5*x2*y3,
.
.
.
    x0^2*x1^3*y3 + 3*x1^5*y3 + 6*x1^4*x2*y3 + 5*x0^2*x1*x2^2*y3 + 6*x0*x1^2*x2^2*y3
+ x1^3*x2^2*y3 + 2*x0^2*x2^3*y3
        + x0*x2^4*y3 + 6*x1*x2^4*y3 + 6*x2^5*y3,
    x0*x1^4*y3 + 5*x1^5*y3 + 6*x1^4*x2*y3 + 5*x0*x1^2*x2^2*y3 + 6*x1^3*x2^2*y3
+ 3*x0^2*x2^3*y3 + 4*x0*x1*x2^3*y3 +
        5*x1^2*x2^3*y3 + 6*x0*x2^4*y3 + 2*x1*x2^4*y3 + 5*x2^5*y3,
```

```
    x1^4*x2*y3 + 2*x0^2*x1*x2^2*y3 + 5*x0*x1^2*x2^2*y3 + 3*x0*x1*x2^3*y3 + 6*x1^2*x2^3*y3
+ 2*x0*x2^4*y3 +
        6*x2^5*y3
]
```

Definition 4.11.9 (Affine bilinear system)**.** *The system of polynomials $f_1, \ldots, f_s$ in*

$$k[x_1, \ldots, x_n, y_1, \ldots, y_m]$$

is an affine bilinear system if there exist homogeneous $g_1, \ldots, g_s$ in $k[x_1, \ldots, x_{n+1}, y_1, \ldots, y_{+1}]$ such that

$$f_i(x_1, \ldots, x_n, y_1, \ldots, y_m) = g_i(x_1, \ldots, x_n, 1, y_1, \ldots, y_m, 1)$$

Theorem 4.11.10 (Regularity bound in the affine case)**.** *For most affine bilinear systems, $d_{reg} \leq 1 + \min\{n, m\}$.*

Proof. See [FSEDS11] theorem 6. □

4.11.1 A further decomposition

Write $A_{n,d,e}$ for the vector space of polynomials in A_n having bidegree (d, e). Given an ideal $I \subseteq A_n$ we can consider the polynomials of bidegree (d, e) in it. This set forms a vector space which we denote by $I_{d,e}$. In other words, $I_{d,e} = I \cap A_{n,d,e}$. Recall that we can write

$$I = \bigoplus_{k \in \mathbb{N}} I_k$$

Now, if I is generated by bihomogeneous polynomials we can further decompose I_k. Indeed,

$$I_k = \bigoplus_{\substack{d,e \in \mathbb{N} \\ d+e=k}} I_{d,e}$$

What does this mean from an algorithmic point of view? In [FSEDS11] the authors mention that time is often not the bottleneck when trying to solve a system of equations, but rather memory is. A direct consequence of the above decomposition is then that we are able to decompose the problem of computing a linear basis for I_k into a number of subproblems, namely the computation of linear bases for $I_{d,e}$ with $d + e = k$, the union of which gives us a linear basis for I_k.

How does this translate into an algorithm? Consider the matrix M_k appearing during the execution of the Matrix-F5 algorithm and recall that a linear basis for I_k is obtained by computing an echelon form of M_k. Replace M_k by the matrices $M_{d,e}$ such that $d + e = k$ and the rows of which are indexed by the polynomials of bidegree (d, e), and the columns of which are indexed by all monomials in $A_{n,d,e}$. A linear basis of I_k is now obtained by computing echelon forms of these much smaller matrices, a problem which is a lot more tractable, and possibly lends itself to parallelization.

This idea is readily generalized to systems of multihomogeneous polynomials.

Chapter 5

State of the art: the GVW algorithm

We now present the GVW algorithm, a relatively modern signature-based algorithm which is competitive with Faugère's F5 algorithm, but is conceptually not much different from Buchberger's algorithm. We decided to include it here for its simplicity and for its novel approach to Gröbner basis computation. Like Matrix-F5, the algorithm is signature-based, but it does not make use of linear algebra, although it can be adapted to perform the reductions by matrix operations.

One of the fundamental differences between the GVW algorithm and the F5-family is that it is not incremental in nature. Moreover, the signature order is not fixed and changing this order can have a big impact on the performance of the algorithm. GVW is actually the generalised version of an incremental algorithm called G2V. Toy implementations of both algorithm can be found at the end of this text.

5.1 Theoretical foundations

Our exposition will closely follow [GVIW16]. Recall that A_n is the polynomial ring in the variables $x_1, \ldots, x_n$ over some field k. Let $f_1, \ldots, f_m$ be polynomials in A_n generating an ideal $I = \langle f_1, \ldots, f_m \rangle$. Recall the surjective homomorphism

$$\phi : A_n^m \to I$$

given by

$$\sum_{i=1}^{m} u_i e_i \mapsto \sum_{i=1}^{m} u_i f_i$$

where e_i is the canonical basis of the free module A_n^m. The kernel H of this homomorphism is the A_n-module of all syzygies on $f_1, \ldots, f_m$, i.e.,

$$H = \{(u_1, \ldots, u_m) \in A_n^m : \sum_{i=1}^{m} u_i f_i = 0\}$$

In addition to computing a Gröbner basis for I, the GVW-algorithm computes a Gröbner basis for this syzygy module. The F5 algorithm tries to predict useless zero reductions by

considering only the principal syzygies. The GVW algorithm encounters many non-principal syzygies while trying to compute a basis and uses these to predict useless zero reductions.

We consider the submodule

$$M = \{(u, f) \in A_n^m \times A_n : \phi(u) = f\} \subset A_n^m \times A_n$$

generated by

$$(e_1, f_1), \ldots, (e_m, f_m)$$

that allows us to simultaneously operate on both elements from H and I.

In the following we make the convention that $\mathrm{lm}(f) = 0$ if $f = 0$ and $\mathrm{lm}(u) = 0$ if $u = 0$. Moreover, we will need a map $A_n^m \times A_n^m \to A_n$ which we will call division. This map takes two terms, i.e., a vector of the form $x^\alpha e_i$ where e_i is the ith unit vector, and is only defined if $i = j$ and x^α divides x^β in the usual sense. The result will be the monomial $x^{\beta-\alpha} \in R$. In this case, we will say that $x^\alpha e_i$ divides $x^\beta e_j$.

Definition 5.1.1 (Signature)**.** *Let $(u, f) \in A_n^m \times A_n$. The signature of the pair (u, f) is defined as the signature of u, i.e., $\sigma(u, f) = \sigma(u) = \mathrm{lm}(u)$.*

Definition 5.1.2 (Top-reducibility)**.** *Let $p_1 = (u, f)$ and $p_2 = (v, g)$ be any two pairs in $A_n^m \times A_n$.*

- *If $g \neq 0$, we say that p_1 is top-reducible by p_2 if*
  - *$f \neq 0$ and $\mathrm{lm}(g)$ divides $\mathrm{lm}(f)$, and*
  - *$\mathrm{lm}(tv) \leq \mathrm{lm}(u)$ where $t = \frac{\mathrm{lm}(f)}{\mathrm{lm}(g)}$.*

  The corresponding top-reduction is $(u, f) - ct(v, g) = (u - ctv, f - ctg)$ where $c = \frac{\mathrm{lc}(f)}{\mathrm{lc}(g)}$.
- *If $g = 0$, we say that p_1 is top-reducible by p_2 if $\mathrm{lm}(v)$ divides $\mathrm{lm}(u)$. The corresponding top-reduction is $(u, f) - ct(v, 0) = (u - ctv, f)$ where $c = \frac{\mathrm{lc}(u)}{\mathrm{lc}(v)}$ and $t = \frac{\mathrm{lm}(u)}{\mathrm{lm}(v)}$.*

Definition 5.1.3 (Regular top-reducibility)**.** *A top-reduction is called regular if $\mathrm{lm}(u) > \mathrm{lm}(tv)$.*

Definition 5.1.4 (Super top-reducibility)**.** *A top-reduction is called super if it is not regular, i.e., if $\mathrm{lm}(u) = \mathrm{lm}(tv)$*

Observe that a pair is super top-reducible if and only if

$$\mathrm{lm}(tu) = \mathrm{lm}(v) \text{ and } \frac{\mathrm{lc}(u)}{\mathrm{lc}(v)} = \frac{\mathrm{lc}(f)}{\mathrm{lc}(g)}$$

The key difference between the two notions of reducibility is that the signature remains invariant under a regular top-reduction whereas it becomes smaller after a super top-reduction. The algorithm never performs super top-reductions as they do not yield any new information.

Definition 5.1.5. *Let Let $G \subset A_n^m \times A_n$. A pair p is called regular top-reducible by G if there exists a pair q in G such that p is regular top-reducible by q.*

Definition 5.1.6 (Strong Gröbner basis)**.** *A subset G of M is called a strong Gröbner basis for M if every nonzero pair in M is top-reducible by some pair in G.*

Recall that as a submodule M naturally has a Gröbner basis. In general, the notion of a strong Gröbner basis given above does not coincide with the former, although they are similar. Both are saying that every element of the submodule is top-reducible by some element from the Gröbner basis. The difference is that the top-reductions defined in this chapter are signature-preserving.
From a strong Gröbner basis we can extract Gröbner bases for the ideal I and its syzygy module H as follows:

Proposition 5.1.7. *Let $G = \{(u_1, v_1), (u_2, v_2), \ldots, (u_t, v_t)\}$ be a strong Gröbner basis for M. Then*

- *$G_0 = \{u_i : v_i = 0, 1 \leq i \leq t\}$ is a Gröbner basis for the syzygy module of $f_1, \ldots, f_m$, and*
- *$G_1 = \{v_i : 1 \leq i \leq t\}$ is a Gröbner basis for the ideal $\langle f_1, \ldots, f_m \rangle$.*

Proof. First, we prove that G_0 is a Gröbner basis for the syzygy module of $f_1, \ldots, f_m$. To this end, let u be a syzygy on $f_1, \ldots, f_m$, then $(u, 0) \in M$. Since G is a strong Gröbner basis for M it follows that $(u, 0)$ is top-reducible by (u_i, v_i) for some $1 \leq i \leq t$. By definition of top-reducibility, we have that $v_i = 0$ and $\mathrm{lm}(u_i)$ divides $\mathrm{lm}(u)$. Moreover, $(u_i, v_i) \in G_0$. From this the result follows. Next, we will prove that G_1 is a Gröbner basis for I. To this end, let $0 \neq v \in I$. Then there exists $u = (u_1, \ldots, u_m) \in A_n^m$ such that $\phi(u_1, \ldots, u_m) = v$. Let u be minimal with respect to this property. It follows that $(u, v) \in M$. Since G is a strong Gröbner basis for M there exists an $1 \leq i \leq t$ such that (u, v) is top-reducible by (u_i, v_i). If $v_i = 0$, then we can reduce (u, v) by $(u_i, 0)$ to get an element $(u', v) \in M$ with $\phi(u') = v$ and $u' < u$, which is a contradiction. If $v_i \neq 0$, then by definition $\mathrm{lm}(v)$ is divisible by $\mathrm{lm}(v_i)$. Hence G_1 is a Gröbner basis for I. □

Next, we define the notion of a joint pair, abbreviated as J-pair:

Definition 5.1.8 (J-pair)**.** *Let $p_1 = (u, f)$ and $p_2 = (v, g)$ be two pairs from $A_n^m \times A_n$ with both f and g nonzero. Let*

$$t_f = \frac{\mathrm{lcm}(\mathrm{lm}(f), \mathrm{lm}(g))}{\mathrm{lm}(f)}, \quad t_g = \frac{\mathrm{lcm}(\mathrm{lm}(f), \mathrm{lm}(g))}{\mathrm{lm}(g)}$$

Suppose that $\max\{t_f \,\mathrm{lm}(u), t_g \,\mathrm{lm}(v)\} = t_f \,\mathrm{lm}(u)$*. Then*

- *The J-pair of p_1 and p_2 equals $t_f p_1 = (t_f u, t_f f)$.*
- *The J-signature of p_1 and p_2 equals $t_f \,\mathrm{lm}(u)$.*

Remark 17. When $t_f \,\mathrm{lm}(u) = t_g \,\mathrm{lm}(v)$ the J-pair of (u, f) and (v, g) remains undefined. This condition is easily seen to be equivalent to $\mathrm{lm}(g)\,\mathrm{lm}(u) = \mathrm{lm}(f)\,\mathrm{lm}(v)$ since $\mathrm{lcm}(\mathrm{lm}(f), \mathrm{lm}(g)) \neq 0$. The latter condition can easily be computed without having to compute the lcm, hence we use it as a means of checking whether we need to compute a J-pair in the algorithm.

The definition should remind the reader of the S-polynomial in Buchberger's algorithm. The difference between the two concepts is that the leading term of both polynomials is cancelled during the construction of the S-polynomial, whereas the J-pair postpones such a cancellation. By construction, the J-pair of two pairs p_1 and p_2 is always top-reducible by either p_1 or p_2. However, it may happen that there exists a pair p_3 distinct from p_1 and p_2 which is a better reductor. We are being intentionally vague about what it means to be better since this can only be backed up by empirical evidence.

Definition 5.1.9 (Eventually super top-reducible)**.** *Let $G \subset A_n^m \times A_n$. A pair p is called eventually super top-reducible by G if there exists a sequence of regular top-reductions of p by pairs in G such that p reduces to a pair p' that is no longer regular top-reducible by G, but there exists a pair q in G such that p' is super top-reducible by q.*

Definition 5.1.10 (Covered)**.** *A pair p_1 is covered by another pair p_2 if $\sigma(p_2)$ divides $\sigma(p_1)$ and $t\operatorname{lm}(v_2) < \operatorname{lm}(v_1)$ where $t = \frac{\operatorname{lm}(u_1)}{\operatorname{lm}(u_2)}$. We also say that p_2 covers p_1.*

Definition 5.1.11 (Covered by G)**.** *Let $G \subset A_n^m \times A_n$. A pair p is covered by G if there exists a pair q in G such that p is covered by q.*

Recall that Buchberger's algorithm is based on a simple test, Buchberger's criterion. In this new framework, an analogous test exists for determining whether a given set constitutes a strong Gröbner basis without the need for reducing any polynomials. First, we need a lemma.

Lemma 5.1.12. *Let $t \in T_n$ and $p = (u, f), q = (v, g) \in A_n^m \times A_n$. If tp is regular top-reducible by q, then $t_1 p$ is a J-pair of p and q where*

$$t_1 = \frac{\operatorname{lcm}(\operatorname{lm}(f), \operatorname{lm}(g))}{\operatorname{lm}(f)} = \frac{\operatorname{lm}(g)}{\gcd(\operatorname{lm}(f), \operatorname{lm}(g))} \text{ and } t_1 \text{ divides } t$$

In addition to this, $t_1 p$ is regular top-reducible by q.

Proof. Since tp is regular top-reducible by q we see that both f and g are nonzero. Moreover, there exists an $s \in T$ such that

$$t\operatorname{lm}(f) = s\operatorname{lm}(g) \text{ and } t\operatorname{lm}(u) = \operatorname{lm}(tu - csv) \tag{5.1}$$

where $c = \frac{t\operatorname{lm}(f)}{\operatorname{lm}(g)}$. Let

$$t_2 = \frac{\operatorname{lcm}(\operatorname{lm}(f), \operatorname{lm}(g))}{\operatorname{lm}(g)} = \frac{\operatorname{lm}(f)}{\gcd(\operatorname{lm}(f), \operatorname{lm}(g))}$$

The first equation of 5.1 implies that

$$t = \frac{\operatorname{lm}(g)}{\gcd(\operatorname{lm}(f), \operatorname{lm}(g))} w = t_1 w$$

and

$$s = \frac{\operatorname{lm}(f)}{\gcd(\operatorname{lm}(f), \operatorname{lm}(g))} w = t_2 w$$

for some $w \in T$. The second equation of 5.1 then implies that $t_1 \operatorname{lm}(u) = \operatorname{lm}(t_1 u - c t_2 v)$. This shows that $t_1 p$ is the J-pair of p and q, and $t_1 p$ is regular top-reducible by q. □

Next, we present the theorem on which the test is based.

Theorem 5.1.13. *Suppose that G is a subset of M such that for any term $T \in A_n^m$, there exists a pair $(u, f) \in G$ such that $\mathrm{lm}(u)$ divides T. The following are equivalent:*

1. *G is a strong Gröbner basis for M.*
2. *Every J-pair is eventually super top-reducible by G.*
3. *Every J-pair of G is covered by G.*

Proof. (1) $\Rightarrow$ (2): Let $(u, f) \in M$ be a J-pair. Since G is a strong Gröbner basis for M there exists a pair in G top-reducing (u, f). We may perform a sequence of such top-reductions to end up with a pair $(v, g) \in M$ that is no longer regular top-reducible. Since $(v, g) \in M$ it is again top-reducible by G, but it is not regular top-reducible, so it must be super top-reducible. This is precisely what it means for a pair to be eventually super top-reducible.
(2) $\Rightarrow$ (3): Let $(u, f) \in G$ be a J-pair. By (2) it is eventually super top-reducible by G, i.e., there exists a sequence of regular top-reductions ending in a pair $(v, g) \in G$ such that (v, g) is no longer regular top-reducible, but is super top-reducible by a pair $(w, h) \in G$. Since a J-pair is regular top-reducible by construction, we deduce that $\mathrm{lm}(g) < \mathrm{lm}(f)$ and $\mathrm{lm}(u) = \mathrm{lm}(v)$. If $h = 0$, then $\mathrm{lm}(w)$ divides $\mathrm{lm}(v) = \mathrm{lm}(u)$ and $t\,\mathrm{lm}(h) = 0 < \mathrm{lm}(f)$, so (u, f) is covered by (w, h). If $h \neq 0$, then

$$\mathrm{lm}(tw) = \mathrm{lm}(v)$$

where $t = \frac{\mathrm{lm}(g)}{\mathrm{lm}(h)}$. It follows that $t = \frac{\mathrm{lm}(u)}{\mathrm{lm}(w)}$. Then $t\,\mathrm{lm}(w) = \mathrm{lm}(u)$, i.e., $\mathrm{lm}(w)$ divides $\mathrm{lm}(u)$, and $t\,\mathrm{lm}(h) = \mathrm{lm}(g) < \mathrm{lm}(f)$. It follows that (u, f) is covered by (w, h).
(3) $\Rightarrow$ (1): We prove by contradiction. Suppose that G is not a strong Gröbner basis for M. Then there exists a pair $p = (u, f) \in M$ that is not top-reducible by any pair in G. Among the pairs satisfying this property we pick one with minimal signature $\mathrm{lm}(u)$. We remark that $\mathrm{lm}(u) \neq 0$, since the pair $(0, 0)$ is always super top-reducible. Next, we pick a pair $q = (v, g)$ from G satisfying

1. $\mathrm{lm}(u) = t\,\mathrm{lm}(v)$ for some $t \in T_n$. The existence of such a pair follows from the assumption stated in the theorem.
2. $t\,\mathrm{lm}(g)$ is minimal among all q satisfying (i).

We claim that tq is not regular top-reducible by G. We prove this by exhibiting a contradiction to property (ii). Suppose that tq is regular top-reducible by G, say by $\tilde{q} = (\tilde{v}, \tilde{g}) \in G$. Note that this implies that both g and $\tilde{g}$ are nonzero. By lemma 5.1.12 the J-pair of q and $\tilde{q}$ equals $t_1 q$ where

$$t_1 = \frac{\mathrm{lcm}(\mathrm{lm}(g), \mathrm{lm}(\tilde{g}))}{\mathrm{lm}(g)} \text{ and } t = t_1 w$$

for some $w \in T_n$. By assumption (3) the J-pair $t_1 q$ is covered by G, i.e., there exists a pair $\bar{q} = (\bar{v}, \bar{g}) \in G$ with $t_2\,\mathrm{lm}(\bar{g}) < t_1\,\mathrm{lm}(g)$ where $t_2 = \frac{t_1\,\mathrm{lm}(v)}{\mathrm{lm}(\bar{v})}$. Then

$$\mathrm{lm}(u) = t\,\mathrm{lm}(v) = t_1 w\,\mathrm{lm}(v) = t_2 w\,\mathrm{lm}(\bar{v})$$

and

$$w t_2\,\mathrm{lm}(\bar{g}) < t_1 w\,\mathrm{lm}(g) = t\,\mathrm{lm}(g)$$

which contradicts property (ii).

Now, consider the following reduction

$$(\bar{u}, \bar{f}) = p - ctq = (u, f) - ct(v, g)$$

where $c = \frac{\mathrm{lc}(u)}{\mathrm{lc}(v)}$. By construction, we have that $\mathrm{lm}(\bar{u}) < \mathrm{lm}(u)$. Since we assumed that p is not top-reducible we see that $\mathrm{lm}(f) \neq t\,\mathrm{lm}(g)$. Moreover, $(\bar{u}, \bar{f})$ is top-reducible by G. For if it were not, then we would derive a contradiction based on the facts that $(\bar{u}, \bar{f}) \in M$, $\mathrm{lm}(\bar{u}) < \mathrm{lm}(u)$, and recalling that (u, f) was chosen to be minimal with respect to this property. Hence there exists a pair $r = (w, h) \in G$ top-reducing $(\bar{u}, \bar{f})$. Without loss of generality, we may assume that $h \neq 0$ (Indeed, just keep reducing by pairs with h-part equal to 0 until this holds). Recall that $\mathrm{lm}(f) \neq t\,\mathrm{lm}(g)$, so we can consider two cases:

- $\mathrm{lm}(f) < t\,\mathrm{lm}(g)$. Then $\mathrm{lm}(\bar{f}) = t\,\mathrm{lm}(g)$, hence tq is regular top-reducible by r, contradicting the assumption that it was not.
- $\mathrm{lm}(f) > t\,\mathrm{lm}(g)$. Then $\mathrm{lm}(\bar{f}) = \mathrm{lm}(f)$. Now, since $(\bar{u}, \bar{f})$ is top-reducible by r we see that (u, f) is top-reducible by r, contradicting the assumption that it was not.

Since we arrive at a contradiction in both cases, such a pair p does not exist. It follows that every pair in M is top-reducible by G. This is precisely the condition for G to be a strong Gröbner basis for M. □

This simple test automatically yields two criteria for predicting superfluous calculations. For the first criterion, observe that a J-pair is covered by a syzygy if and only if it is top-reducible by this syzygy. This leads to the following syzygy criterion:

Corollary 5.1.14 (Syzygy criterion)**.** *If a J-pair is top-reducible by a syzygy, then it may be discarded.*

Now, if p_1 covers p_2 and p_2 covers p_3, then p_1 covers p_3. It follows that the covering relation is transitive. From this fact we deduce the following criterion:

Corollary 5.1.15 (Signature criterion)**.** *If a J-pair is covered by a pair in G or covered by another J-pair, then it may be discarded.*

The signature criterion shows that, given a signature, we only need to store one J-pair with that signature. Experiments show that choosing the J-pair with minimal f-part performs best, but this is just a heuristic.

5.2 The algorithm

As remarked, the algorithm has the same flavor as Buchberger's algorithm. It is based on the characterization provided by theorem 5.1.13. In particular, we will incrementally compute new J-pairs of pairs in G until every single one of them is covered by G, starting with the initial pairs $(e_1, f_1), \ldots, (e_m, f_m)$. In the actual implementation, we store these pairs in two lists, U and V, where U will store the u-part and V will store the f-part of the pair (u, f). Once the algorithm terminates, V will contain a Gröbner basis for $\langle f_1, \ldots, f_m \rangle$. Since it is

expensive to work with the vectors (u, f) we keep the u-part monic and only store $(\mathrm{lm}(u), f)$. This doesn't affect the outcome of the algorithm if we are only interested in a Gröbner basis for $\langle f_1, \ldots, f_m \rangle$. If, however, we are also interested in a Gröbner basis for the syzygy module of $f_1, \ldots, f_m$ then we do need to store the entire vector representation. Once the algorithm terminates, the polynomials in U will form a basis for this syzygy module. Now, let (u, f) and (v, g) be any two pairs and suppose that we only store $(\mathrm{lm}(u), f)$ and $(\mathrm{lm}(v), g)$. Then $(\mathrm{lm}(u), f)$ is regular top-reducible by $(\mathrm{lm}(v), g)$ when $g \neq 0$, $\mathrm{lm}(f)$ is divisible by $\mathrm{lm}(g)$ and $t\,\mathrm{lm}(v) < \mathrm{lm}(u)$ or $t\,\mathrm{lm}(v) = \mathrm{lm}(u)$, but $\mathrm{lc}(f) \neq \mathrm{lc}(g)$. The corresponding top-reduction then is

$$h := f - ctg$$

where $t = \frac{\mathrm{lm}(f)}{\mathrm{lm}(g)}$ and $c = \frac{\mathrm{lc}(f)}{\mathrm{lc}(g)}$, and if $t\,\mathrm{lm}(v) = \mathrm{lm}(v)$, then we update v as

$$v = \frac{v}{1 - c}$$

as to keep the u-part of (u, h) monic. The result of the top-reduction is the pair $(\mathrm{lm}(u), h)$. All the regular top-reductions performed by the algorithm are of this form.

For every system we are interested in, we know a number of relations between the generators in advance. For example, we always know that the relations coming from the commutativity of A_m hold. Moreover, we have seen in the section on the various extensions of Matrix-F5 that sometimes we know more than just these relations. Therefore, it makes sense to store the leading terms of the principal syzygies (and any other syzygies we know) in a list H which will be updated every time we encounter a new zero reduction and will be used to predict future zero reductions. Moreover, any time we add a new pair to the list of pairs found so far, we also add the trivial relations between that pair and the known pairs to H.

Theorem 5.2.1 (Finite termination and correctness)**.** *Assume that the monomial order on A_n is compatible with the monomial order on A_n^m. Then algorithm 14 terminates in a finite number of steps and outputs a strong Gröbner basis for M.*

Proof. First, we prove the finite termination. Given two pairs $p = (u, f), q = (v, g) \in R^m \times R$ we say that p divides q if $\mathrm{lm}(u)$ divides $\mathrm{lm}(v)$ and $\mathrm{lm}(f)$ divides $\mathrm{lm}(g)$. Consider the list of pairs in G where a pair occurs before another pair if the algorithm inserted the former pair first:

$$(e_1, f_1), \ldots, (e_m, f_m), (\mathrm{lm}(u_1), g_1), (\mathrm{lm}(u_2), g_2), \ldots$$

Put $p_i = (u_i, g_i)$ for $i \geq 1$. We claim that p_i does not divide p_j whenever $i < j$. To see why this is, we reason by contradiction. Suppose that p_i divides p_j for some $i < j$. Then

$$\mathrm{lm}(u_j) = s\,\mathrm{lm}(u_i) \text{ and } \mathrm{lm}(g_j) = t\,\mathrm{lm}(g_i)$$

for certain $s, t \in T_n$.

We now consider two cases:

- If $s > t$, then $\mathrm{lm}(u_j) = s\,\mathrm{lm}(u_i) > t\,\mathrm{lm}(u_i)$. It follows that p_i regular top-reduces p_j. However, p_i is not regular top-reducible, as it is the result of a sequence of regular top-reductions until it no longer is. This is a contradiction.

input : $F = \{f_1, \ldots, f_m\}$ a sequence of polynomials and monomial orders $\leq$ and $\leq$ on A_n and A_n^m respectively
output: A Gröbner basis for $\langle F \rangle$ with respect to $\leq$
begin
  $U := \{e_1, \ldots, e_m\}$;
  $V := F$;
  $H := \{\mathrm{lm}(f_j e_i - f_i e_j) : 1 \leq i < j \leq m\}$;
  $JP :=$ `Sort(`$\{$*J-pairs of* $(e_1, f_1), \ldots, (e_m, f_m)\}$`)` ;
  while $JP \neq \emptyset$ **do**
    $(x^\alpha, i) :=$ the first element of JP;
    $JP :=$ `Sort(`$JP \setminus \{(x^\alpha, i)\}$`)` ;
    if $x^\alpha T_i$ *is divisible by some monomial from* H **then**
      continue;
    end
    if $x^\alpha(T_i, v_i)$ *is covered by* (U, V) **then**
      continue;
    end
    $(T, v) :=$ result of regular top-reduction of $x^\alpha(T_i, v_i)$ by (U, V) ;
    if $v = 0$ **then**
      $H := H \cup \{T\}$;
      continue;
    end
    for $j := 1$ **to** $|U| - 1$ **do**
      if $v_j T \neq v T_j$ **then**
        $H := H \cup \max\{v_j T, v T_j\}$;
        $JP := JP \cup \{$`JPair(`$(T, v), (T_j, v_j)$`)`$\}$;
      end
    end
    $U := U \cup \{T\}$;
    $V := V \cup \{v\}$;
  end
  return V
end

Algorithm 14: GVW algorithm

input : $G = \{g_1, \ldots, g_s\}$ a Gröbner basis for $I = \langle f_1, \ldots, f_i \rangle$, and f_{i+1} a polynomial.
Monomial orders $\leq$ and $\leq$ on A_n and A_n^m respectively
output: A Gröbner basis for $I + f_{i+1}$ with respect to $\leq$
begin
  $U := \{0, \ldots, 0\}$ (s times) ;
  $V := G$;
  $H := \{\mathrm{lm}(g_i) : 1 \leq i \leq s\}$;
  $v := f_{i+1} \operatorname{rem} G$;
  if $v = 0$ **then**
    $H := H \cup \{1\}$;
    return V ;
  end
  $U := U \cup \{1\}$;
  $V := V \cup \{v\}$;
  $JP :=$ `Sort(`$\{$*J-pairs of* $(0, g_1), \ldots, (0, g_s)$ *and* $(1, v)$`)` ;
  while $JP \neq \emptyset$ **do**
    $(x^\alpha, i) :=$ the first element of JP;
    $JP :=$ `Sort(`$JP \setminus \{(x^\alpha, i)\}$`)` ;
    if $x^\alpha T_i$ *is divisible by some monomial from* H **then**
      continue;
    end
    if $x^\alpha(T_i, v_i)$ *is covered by* (U, V) **then**
      continue;
    end
    $(T, v) :=$ result of regular top-reduction of $x^\alpha(T_i, v_i)$ by (U, V) ;
    if $v = 0$ **then**
      $H := H \cup \{T\}$;
      continue;
    end
    for $j := 1$ **to** $|U| - 1$ **do**
      if $v_j T \neq v T_j$ **then**
        $H := H \cup \max\{v_j T, v T_j\}$;
        $JP := JP \cup \{$`JPair(`$(T, v), (T_j, v_j)$`)`$\}$;
      end
    end
    $U := U \cup \{T\}$;
    $V := V \cup \{v\}$;
  end
  return V
end

Algorithm 15: G2V algorithm

- If $s \leq t$, then $s\,\mathrm{lm}(g_i) \leq t\,\mathrm{lm}(g_i) = \mathrm{lm}(g_j)$. Let $p = (u, f)$ be the J-pair that was reduced to p_j by the algorithm. Since a J-pair is always regular top-reducible we deduce that $s\,\mathrm{lm}(g_i) \leq \mathrm{lm}(g_j) < \mathrm{lm}(f)$ and $\mathrm{lm}(u) = \mathrm{lm}(u_j) = s\,\mathrm{lm}(u_i)$. It follows that p_i covers p. This is a contradiction, as this would imply that p was discarded at the covering check.

Hence p_i does not divide p_j whenever $i < j$. So we end up with a sequence

$$(\mathrm{lm}(u_1), g_1), (\mathrm{lm}(u_2), g_2), \ldots$$

in which no term is divisible by any previous term. Now, consider the homomorphism

$$h : A_n^m \times A_n \to k[y_1, \ldots, y_m, x], \quad (x^\alpha e_i, x^\beta) \mapsto y_i^\alpha x^\beta$$

where $y_i = (y_1, \ldots, y_n)$ for $1 \leq i \leq m$, and $x = (x_1, \ldots, x_n)$. Next, define $I_1 = \langle h(\mathrm{lm}(u_1), g_1)\rangle$ and $I_{i+1} = I_i + \langle h(\mathrm{lm}(u_{i+1}), g_{i+1})\rangle$. This gives us an ascending chain in $k[y_1, \ldots, y_m, x]$:

$$I_1 \subseteq I_2 \subseteq \cdots$$

which eventually stabilizes to due to the Noetherianity of $k[y_1, \ldots, y_m, x]$. It follows that G is a finite set.

Next, let G be the output of the algorithm (recall that G is given by U and V). We prove that G is indeed a strong Gröbner basis for M. By theorem 5.1.13, G is a strong Gröbner basis for M if every J-pair of G is covered by G. Suppose, for the sake of contadiction, that there exists a J-pair of G, say (u, f), that is not covered by G. Then this J-pair is extracted from JP at some point in the algorithm and is top-reduced by a sequence of regular top-reductions (at least one!) to another pair (u, g) with $\mathrm{lm}(g) < \mathrm{lm}(f)$. But then (u, g) covers (u, f) and (u, g) is added to G by the algorithm. This is a contradiction. □

5.3 Complexity

Unfortunately, not much seems to be known about the complexity of the GVW algorithm other than that it is comparable to that of the F5-family. Therefore, studying the complexity of the GVW algorithm reduces to the problem of finding out what the highest degree is that is reached during the computation of a Gröbner basis.

Chapter 6

Experimental results

In this chapter we will discuss a small number of experiments. Our toy implementations do not compete with the implementation of F4 that is present in the Magma computer algebra system, although our implementation of G2V behaves nicely, considering that it is not completely optimized. The running time of Matrix-F5 relies heavily on the computation of a particular type of echelon form, i.e., one that is the result of Gaussian elimination without pivoting. Our implementation computes this in a very naive way, which is quite slow. We tried to use Magma's built-in echelon form method. Unfortunately, we were not able to extract the permutation matrix from the transformation matrix that is returned by the method, which is what we need to permute back the rows in order to get the desired echelon form. We believe that a hybrid between Matrix-F5 and the F4 algorithm could be very competitive. This has been considered by, e.g., Albrecht in [Alb10].

The first experiment shows the absolute running times of the various algorithms on some well known polynomial systems. A horizontal bar denotes that the computation took too much time, so we aborted it. Note that it is not fair to compare the algorithms in this way, as the running time depends very much on the implementation. As we have said, Magma's implementation is highly optimized, whereas our implementation of Matrix-F5 is far from optimal. It would be interesting to see how Matrix-F5 performs when the modified Gaussian elimination algorithm is optimized. Also, we feel that G2V might be able to beat Magma's implementation if the reduction step were implemented with fast linear algebra similar to what F4 does.

	Time(s)			
	G2V	Magma	Matrix-F5	Modified Matrix-F5
Cyclic(4)	0.000	0.000	17.970	17.030
Cyclic(5)	15.210	0.000	-	-
Cyclic(6)	-	0.030	-	-
Katsura(4)	0.030	0.000	42.610	42.590
Katsura(5)	70.130	0.030	-	-
Katsura(6)	-	1.750	-	-

For the second experiment we computed a number of invariants of some homogeneous systems that we encountered during the writing of this thesis. Unfortunately, there does not seem to

be an obvious relation between the quantities.

d_{reg}	$\dim I$	i_{reg}
4	1	3
6	4	4
6	2	5
4	1	3
3	1	6
4	1	6
3	1	6
5	2	5

In the last experiment we looked at the system from Example 4.11.8. As expected, both the modified Matrix-F5 algorithm and the Matrix-F5 algorithm extended to the bilinear case perform much better than the original Matrix-F5 algorithm. It is interesting to see that the modified Matrix-F5 algorithm does not perform much worse than the version of Matrix-F5 which was optimized for systems of this particular shape.

	Matrix-F5	Modified Matrix-F5	MatrixF5Bilinear
Time (s)	107.410	99.010	98.420

Part II

Algebraic coding theory

Chapter 7

Basic concepts of linear codes

In many applications one wants to transfer information from one party to another. The sending party will be referred to as the sender and the receiving party as the receiver. The channel through which this information is sent is often noisy and as a result errors may occur. Of course, once an error has occurred, the receiver may simply ask the sender to retransmit the data. However, this is not very efficient from a practical point of view. Moreover, sometimes this isn't even possible. The sender adds redundant information to the data being sent in such a way that the original data may be reconstructed from the redundant information if not too many errors take place. The process of adding redundant information is referred to as encoding. The process of retrieving the original message from its distorted encoded form is referred to as decoding. The set of all encoded messages, henceforth called codewords, is called an error-correcting code, or simply code.

We will study a particular class of codes called block codes. The idea here is that the sender encodes data in blocks of a fixed length k with symbols from a fixed alphabet into codewords of a fixed length n with symbols from the same alphabet, and transmits these to the receiver. In addition to this, we will impose a linear algebraic structure on the codes under consideration. The alphabet for a linear code is always a finite field. Linear codes are easier to construct, encode, and decode than their non-linear counterpart.

In this chapter, we will solve the decoding problem by means of Gröbner bases. We accomplish this by rewriting the problem into a system of equations. The solution to these equations is closely related to the errors that have occurred during transmission. However, this problem is, in general, NP-hard. McEliece used this fact to develop a public key cryptosystem for which there are no known effective attacks. In particular, it is known to withstand attacks using Shor's algorithm, making it a candidate for post-quantum cryptography. An immediate consequence is that speed-ups in the decoding algorithm translate into a loss of security.

7.1 Introduction

Definition 7.1.1 (Linear code)**.** *A linear block code C of length n and dimension k is a k-dimensional linear subspace of the n-dimensional vector space over $\mathbb{F}_q$.*

We call n and k the parameters of the code and we will often abbreviate the above by saying that C is an $[n, k]$-code. It is customary in coding theory to view vectors as row vectors.

Vectors in $\mathbb{F}_q^n$ are referred to as words and vectors belonging to C are called codewords.

Definition 7.1.2 (Information rate and redundancy)**.** *The information rate is the quantity $R(C) = \frac{k}{n}$. It is a measure of how much information is being transmitted per codeword. It is closely related to the redundancy $r = n - k$, the number of parity symbols in a codeword.*

As a k-dimensional linear space, C admits a basis for vectors $g_1, \ldots, g_k$ such that any codeword c in C has a unique linear representation with respect to this basis. If we collect these vectors in a matrix, we find a method of describing C in an explicit way. This leads to the notion of a generator matrix.

Definition 7.1.3 (Generator matrix)**.** *A generator matrix of an $[n, k]$-code is a $k \times n$ matrix G whose row space equals C, i.e., such that*

$$C = \{mG : m \in \mathbb{F}_q^k\}.$$

In the case that $G = [I_k|P]$ where I is the $k \times k$ identity matrix and P is an $k \times (n - k)$ matrix we say that G is in standard form. The code is then said to be systematic at the first k positions, or simply systematic. If G is in standard form, the message bits are embedded in the first k positions of its encoded form and we call the encoding function a systematic encoder. Generalizing this, let $J = \{j_1, \ldots, j_k\}$ be any subset of $\{1, \ldots, n\}$ of size k such that the submatrix consisting of the columns of G indexed by J forms the $k \times k$ identity matrix. In this context, we say that C is systematic at J and call J an information set.
On the other hand, we can also describe C implicitly as the nullspace of a system of equations, the parity check equations. This leads to the notion of a parity check matrix.

Definition 7.1.4 (Parity check matrix)**.** *A parity check matrix of an $[n, k]$-code C is an $(n - k) \times n$ matrix H such that*

$$C = \{c \in \mathbb{F}_q^n : Hc^t = 0\}.$$

Given a parity check matrix, it is easy to check if errors have occurred. We simply multiply the received word by H and check whether the result equals zero. A nonzero result implies that an error has occurred.

Associated with a code are a number of invariants, the most important one of which is its minimum distance. Suppose any two distinct codewords differ in at least three positions and let r be a received word originating from some codeword c and distorted by an error vector e, i.e., $r = c + e$. If the number of errors is at most one, then r resembles c more than it resembles any other codeword. Therefore, the receiver will assume c was sent. This idea leads to the following natural metric on the vector space $\mathbb{F}_q^n$.

Definition 7.1.5 (Hamming distance)**.** *The Hamming distance between the words $x, y \in \mathbb{F}_q^n$ is defined to be*

$$d(x, y) = |\{1 \leq i \leq n : x_i \neq y_i\}|.$$

In a similar way, we define the Hamming distance between a word y and a linear code C as

$$d(y, C) = \min\{d(y, c) : c \in C\}.$$

We are now ready to define the minimum distance, the smallest distance between any two distinct codewords.

Definition 7.1.6 (Minimum distance)**.** *Let C be a code with at least two codewords. We define the minimum distance of C to be*

$$d = d(C) = \min\{d(x, y) : x, y \in C, x \neq y\}.$$

For applications one wants to compare codes with different parameters, so it makes sense to look at ratios. To this end, we define the relative distance of a code.

Definition 7.1.7 (Relative distance)**.** *The relative distance of C is the quantity $\delta(C) = \frac{d(C)}{n}$.*

Closely related to the minimum distance is the number of errors that a code can correct. It is referred to as the error-correcting capability.

Definition 7.1.8 (Error-correcting capability)**.** *The error-correcting capability of a linear code C is the quantity $\tau(C) = \left\lfloor \frac{d(C)-1}{2} \right\rfloor$.*

Now, one of the fundamental goals of coding theory is to find codes with a high error-correcting capability, a large relative minimum distance, and a large information rate. Unfortunately, these are conflicting properties. Indeed, the following bound, known as the Singleton bound, immediately makes this clear.

Theorem 7.1.9 (Singleton bound)**.** *Let C be an $[n, k, d]$-code. Then*

$$d \leq n - k + 1$$

Proof. See [MS77] chapter 1, §10, theorem 11. □

Codes that achieve equality in the Singleton bound are called maximum distance separable, or MDS for short.

Definition 7.1.10 (MDS code)**.** *Let C be an $[n, k, d]$-code. If $d = n - k + 1$, then we call C an MDS code.*

MDS codes are important since they have optimal error-correcting capability in the class of all $[n, k]$-codes for fixed n and k.

The number of errors that have occurred is exactly equal to the number of nonzero entries in the error vector. This inspires the following definition:

Definition 7.1.11 (Hamming weight)**.** *The Hamming weight of $y \in \mathbb{F}_q^n$ is defined to be*

$$\mathrm{wt}(y) = d(y, 0).$$

Likewise, we have the Hamming weight of a code C

$$\mathrm{wt}(C) = \min\{d(y, 0) : 0 \neq y \in C\}.$$

Definition 7.1.12 (support)**.** *The support of $y \in \mathbb{F}_q^n$ is the set $\{i : i \in \{1, \ldots, n\}, y_i \neq 0\}$.*

Observe that the weight of a word is exactly equal to the size of its support.

Lemma 7.1.13. *For every $x, y \in \mathbb{F}_q^n$ we have that*

$$\mathrm{wt}(x - y) = d(x, y).$$

Proof.

$$\mathrm{wt}(x-y) = d(x-y,0) = |\{1 \le i \le n : x_i - y_i \neq 0\}| = |\{1 \le i \le n : x_i \neq y_i\}| = d(x,y)$$

□

For linear codes, we have a relation between its minimum distance and its minimum weight.

Proposition 7.1.14. *The minimum distance of a linear code C is equal to* $\mathrm{wt}(C)$.

Proof. Let $x \neq y \in C$. By lemma 7.1.13, we have that $d(x,y) = \mathrm{wt}(x-y)$. Now, $x-y$ is a nonzero vector in C. Taking the minimum on both sides yields the result. □

Let C be an $[n,k]$-code. If, in addition to these parameters, we also know the minimum distance d, we will say that C is an $[n,k,d]$-code.

Proposition 7.1.15. *Let H be a parity check matrix of a linear code C. Then the minimum distance of C is equal to the smallest number d such that there exist d dependent columns of H.*

Definition 7.1.16 (Hamming ball)**.** *The Hamming ball around a word $x \in \mathbb{F}_q^n$ of radius r is the set of all words of distance at most r to x, i.e.,*

$$B(x,r) = \{y \in \mathbb{F}_q^n : d(x,y) \le r\}.$$

Observe that $|B(x,r| = \sum_{i=0}^{r} \binom{n}{i}(q-1)^i$.

7.2 The Golay codes

In this section, we will briefly describe a linear code with special properties that will be used as a running example throughout the remainder of this text: the Golay code. In fact, there are four such codes. The first two such codes are binary.

Definition 7.2.1 (Extended binary Golay code)**.** *The extended binary Golay code, denoted G_{24}, is a $[24,12]$-code over $\mathbb{F}_2$ given by the generator matrix $G = (I_{12}|M)$ in standard form where*

$$M = \begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Now, there are ways of obtaining a new code from a given code. One of those ways is called puncturing. When we puncture a code on a specific coordinate, we simply delete that coordinate from all codewords.

Definition 7.2.2 (Binary Golay code)**.** *The binary Golay code, denoted G_{23}, is a $[23, 12, 7]$ code over $\mathbb{F}_2$ obtained by puncturing G_{24} in any of its coordinates. All these punctured codes are equivalent, so the article "the" is justified.*

Remark 18. Two codes C and C' are said to be equivalent if there exists an isometry

$$\phi : \mathbb{F}_q^n \to \mathbb{F}_q^n$$

such that $\phi(C) = C'$. Recall that an isometry is a map that leaves the (Hamming) metric invariant.

Next there are the two ternary Golay codes.

Definition 7.2.3 (Extended ternary Golay code)**.** *The extended ternary Golay code, denoted G_{12}, is a $[12, 6]$-code over $\mathbb{F}_3$ given by the generator matrix $G = (I_6|M)$ in standard form where*

$$M = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 2 & 1 \\ 1 & 1 & 0 & 1 & 2 & 2 \\ 1 & 2 & 1 & 0 & 1 & 2 \\ 1 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 0 \end{pmatrix}$$

Definition 7.2.4 (Ternary Golay code)**.** *The ternary Golay code, denoted G_{11}, is a $[11, 6, 5]$-code obtained by puncturing G_{12} in any of its coordinates. All these punctured codes are equivalent, so the article "the" is justified.*

Definition 7.2.5 (Perfect code)**.** *A linear code C over $\mathbb{F}_q$ is called perfect if the Hamming balls of radius τ around the codewords cover the entire vector space, i.e., if*

$$\bigcup_{c \in C} B(c, \tau) = \mathbb{F}_q^n$$

where τ denotes the error-correcting capability.

In other words, a perfect code is such that for every word $y \in \mathbb{F}_q^n$ there exists a unique codeword c such that c differs in at most τ positions from y.

Proposition 7.2.6. *Both the binary and ternary Golay codes are perfect.*

7.3 Syndrome decoding

When a receiver receives a word $r = c + e$ composed of a codeword $c \in C$ and an error vector $e \in \mathbb{F}_q^n$, he or she wants to find out what the original codeword was. We now make this precise:

Definition 7.3.1 (Decoding problem)**.** *Let C be a linear code, and let $y \in \mathbb{F}_q^n$ be a received word. The decoding problem is the problem of finding a codeword $c \in C$ such that $d(y, C) = d(y, c)$, i.e., that of finding a codeword closest to y with respect to the minimum distance.*

Berlekamp, et al. [BMvT06] proved that the decoding problem is, in fact, NP-hard. In general, multiple codewords may satisfy the equality above. Therefore, we consider a more restricted problem:

Definition 7.3.2 (Bounded distance decoding problem)**.** *The bounded distance decoding problem is the decoding problem, with the additional assumption that* $d(y, C) \leq \lfloor \frac{d(C)-1}{2} \rfloor$.

This additional assumption ensures that that there exists a unique codeword c satisfying $d(y, C) = d(y, c)$. This is not an unreasonable assumption to make, as the class of perfect codes always satisfies this. It is unknown whether this version of the decoding problem is NP-hard or not, although Vardy [Var97] conjectures that it is. The problem now becomes one of finding an efficient algorithm for retrieving c. A first approach would be to try the method of bruteforce. The receiver simply computes $d(r, c)$ for every $c \in C$ and outputs the c for which $d(r, c)$ is minimal. However, this is not very efficient. In fact, the complexity is equal to $\mathcal{O}(nq^k)$ since there are q^k codewords and for each codeword we have to compute the distance between r and c, which can be done in $\mathcal{O}(n)$ time.
If the information rate $R(C) > \frac{1}{2}$, we can do much better by using the parity check matrix. Recall that r is in the code if $Hr^t = 0$. When e is nonzero, i.e., when an error has occurred, this value will not be equal to zero. This leads to the following definition.

Definition 7.3.3 (Syndrome)**.** *The syndrome of a word* r *with respect to* H *is* $s(r) = Hr^t \in (\mathbb{F}_q)^{n-k}$.

We say that two words x and y are related if their syndromes are equal, i.e., if $s(x) = Hx^t = Hy^t = s(e)$. This happens exactly when $H(x - y)^t = 0$. In other words, x is related to y if their difference $x - y \in C$. This relation is, in fact, an equivalence relation. It follows that the set $\mathbb{F}_q^n$ can be partitioned into disjoint equivalence classes. These equivalence classes are also called cosets. Each coset contains an element of minimal weight, corresponding to the most plausible error pattern. We usually choose this element of minimal weight as a representative and refer to it as the coset leader. Now, a simple calculation shows that the syndromes of both r and e are equal:

$$s(r) = Hr^t = H(c + e)^t = Hc^t + He^t = He^t = s(e)$$

Since r is known to the receiver, the syndrome of e is known to the receiver as well and we assume that this e is in fact the coset leader of the coset of r. It follows that the receiver decodes r as $r - e$. The complexity of this algorithm is $\mathcal{O}(nq^{n-k})$. Assuming that $k > \frac{n}{2}$ this is indeed better than the brute force approach.

input : An $(n - k) \times n$ parity check matrix H describing a linear code C, a vector $r = c + e$ composed of a codeword c and an error vector e of weight at most τ
output: A codeword c
begin
  $s := Hr^t$;
  $e := \min\{\text{wt}(s + c) : c \in C\}$;
  return $r - e$;
end

Algorithm 16: Syndrome decoding

Chapter 8

Cyclic codes

8.1 Introduction

Before returning to the general case, we will study a subclass of linear codes which have good algebraic properties. It turns out that there exist efficient decoding algorithms for decoding such codes.

Definition 8.1.1 (Cyclic code)**.** *A linear code C is said to be cyclic if for every $(c_0, \ldots, c_{n-1}) \in C$ we also have $(c_{n-1}, c_0, \ldots, c_{n-2}) \in C$.*

Given a codeword $c = (c_0, c_1, \ldots, c_{n-1})$ in a cyclic code C, we can associate it with the polynomial $c(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1}$. A cyclic right-shift of a codeword then corresponds to a multiplication by X and taking the remainder after division by $X^n - 1$. In this way, we obtain a one-to-one correspondence between cyclic codes of length n and ideals in the quotient ring $\mathbb{F}_q[X]/(X^n - 1)$. In the following, we will identify the two and write c for the actual codeword and $c(X)$ for its polynomial representation.

Proposition 8.1.2. *A linear code C is cyclic if and only if C is an ideal in $\mathbb{F}_q[X]/(X^n - 1)$.*

Since $\mathbb{F}_q[X]$ is a principal ideal ring it follows that there exists a polynomial $g \in \mathbb{F}_q[X]$ generating C as an ideal.

Definition 8.1.3 (Generator polynomial)**.** *Let C be a non-trivial cyclic code. Then there exists a unique monic polynomial $g \in \mathbb{F}_q[X]$ of lowest degree dividing $X^n - 1$ such that $C = \langle g(X) + \langle X^n - 1 \rangle \rangle$. This polynomial is called the generator polynomial of the cyclic code.*

If we assume that $\gcd(n, q) = 1$, then $X^n - 1$ has no repeated factors over $\mathbb{F}_q$. Let $\mathbb{F}_{q^m}$ be the splitting field of $X^n - 1$, and let $\alpha \in \mathbb{F}_{q^m}^*$ be a primitive nth root of unity. Then $X^n - 1$ factors completely over $\mathbb{F}_{q^m}$ as

$$X^n - 1 = \prod_{i=0}^{n-1} (X - \alpha^i).$$

Since the generator polynomial g divides $X^n - 1$ it follows that

$$g(X) = \prod_{i \in J(C)} (X - \alpha^i)$$

for some subset $J(C) \subseteq \{0, \ldots, n-1\}$. This subset is called the complete defining set of C.

Definition 8.1.4 (Complete defining set)**.** *A complete defining set of C is a subset $J(C)$ of $\{0, \ldots, n-1\}$ comprised of all i such that $c(\alpha^i) = 0$ for all $c \in C$.*

Now, $J(C)$ is partitioned into cyclotomic cosets, i.e., sets of the form $\{q^i j \mod n : i \in \mathbb{N}\}$ for $j \in J(C)$. In order to completely specify the code in an unambiguous way it is sufficient to choose one representative in each coset. This leads to the notion of a defining set.

Definition 8.1.5 (Defining set)**.** *A defining set of C is a subset J of $\{0, \ldots, n-1\}$ such that $c(x) \in C$ if $c(\alpha^i) = 0$ for all $i \in J$.*

A code can have different defining sets. However, there is a unique complete defining set.

8.2 BCH codes

An important class of cyclic codes for which efficient decoding algorithms are known are the BCH codes, named after Bose, Chaudhuri [BRC60], and Hocquenghem [Hoc59].

Definition 8.2.1 (BCH code)**.** *A cyclic code C of length n is said to be a BCH code of designed distance δ if its generator polynomial $g(X)$ is the least common multiple of the minimal polynomials of $\alpha^l, \alpha^{l+1}, \ldots, \alpha^{l+\delta-2}$ for some l, where α is a primitive nth root of unity. This is equivalent with requiring that the complete defining set contains the consecutive integers $l, l+1, \ldots, l+\delta-2$.*

The following lower bound on the minimum distance justifies the use of the word "designed" in the definition above.

Theorem 8.2.2 (BCH bound)**.** *The minimum distance of a BCH code of designed distance δ is at least δ.*

So the theorem states that the minimum distance is least δ if the complete defining set contains $\delta - 1$ consecutive integers.

Example 8.2.3. *The $[11, 6, 5]$ ternary Golay code is equivalent to a cyclic code with complete defining set $J(C) = \{1, 3, 4, 5, 9\}$. Note that we could also have specified the code by the defining set $J = \{1, 3\}$. Its generator polynomial is*

$$g(X) = \prod_{i \in J(C)} (X - \alpha^i) = 2 + X^2 + 2X^3 + X^4 + X^5.$$

Observe that T contains the consecutive integers $3, 4, 5$, hence it is of designed distance $\delta = 4$. The BCH bound then says that $d(C) \geq 4$, but we have already seen that $d(C) = 5$. This shows that the true minimum distance can be strictly larger than what the BCH bound predicts.

8.3 Decoding beyond the BCH error-correcting capability

Throughout the years many algorithms have been developed for decoding BCH codes. We just mention the Peterson-Gorenstein-Zierler decoding algorithm [Pet60] [GZ61], the decoding algorithm by Sugiyama et al. [SKHN75], and the Berlekamp-Massey decoding algorithm [Ber84]. While these algorithms are very efficient in practice, they only correct up to the BCH

error-correcting capability which is derived from the BCH bound. Often the real minimum distance is larger, so we are interested in methods of decoding that incorporate this knowledge. In this section, we will describe a first approach to formulating a system of equations solving the decoding problem in the specific case of a cyclic code. To this end, let C be a τ-error correcting cyclic code with complete defining set $J(C) = \{i_1, \ldots, i_r\}$ with respect to the primitive nth root of unity $\alpha \in \mathbb{F}_{q^m}$. Now, let c be a codeword. We know that its associated polynomial $c(X)$ satisfies $c(\alpha^{i_l}) = 0$ for $1 \leq l \leq r$. Notice that this is the same as saying that $(1, \alpha^{i_l}, \ldots, \alpha^{(n-1)i_l}) \cdot (c_0, \ldots, c_{n-1}) = 0$. It follows that we can use $J(C)$ to define a parity check matrix H for C as follows:

$$H = \begin{pmatrix} 1 & \alpha^{i_1} & \alpha^{2i_1} & \cdots & \alpha^{(n-1)i_1} \\ 1 & \alpha^{i_2} & \alpha^{2i_2} & \cdots & \alpha^{(n-1)i_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{i_r} & \alpha^{2i_r} & \cdots & \alpha^{(n-1)i_r} \end{pmatrix}$$

Now, suppose that the receiver received a word r of the form $r = c + e$ with $c \in C$ and $e \in \mathbb{F}_q^n$. Recall that $s = Hr^t$ is called the syndrome of r with respect to H. The lth entry of s is actually equal to the associated polynomial $r(X)$ evaluated at α^{i_l}. Now, for each $0 \leq i \leq n-1$ we define $s_i = r(\alpha^i)$. For each j in the complete defining set of C we have that $s_j = r(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j)$. We call the s_j with $j \in J(C)$ the known syndromes. The s_j with $j \in \{0, \ldots, n-1\} \setminus J(C)$ are called the unknown syndromes. As the name suggests, the known syndromes are known to the receiver.
If the error vector $e = (e_0, \ldots, e_{n-1})$ is supported by the indices $\{j_1, \ldots, j_t\}$, henceforth referred to as the error positions, then the associated polynomial has the shape

$$e(X) = e_{j_1} X^{j_1} + e_{j_2} X^{j_2} + \cdots + e_{j_t} X^{j_t}.$$

We will call the set $\{e_{j_1}, \ldots, e_{j_t}\}$ the error values. This leads to the following system of equations:

$$\begin{aligned} e_{j_1}(\alpha^{j_1})^{i_1} + \cdots + e_{j_t}(\alpha^{j_t})^{i_1} &= s_{i_1} \\ e_{j_1}(\alpha^{j_1})^{i_2} + \cdots + e_{j_t}(\alpha^{j_t})^{i_2} &= s_{i_2} \\ &\vdots \\ e_{j_1}(\alpha^{j_1})^{i_r} + \cdots + e_{j_t}(\alpha^{j_t})^{i_r} &= s_{i_r} \end{aligned}$$

The set $\{\alpha^{j_1}, \ldots, \alpha^{j_t}\}$ is the set of so-called error locators. Next, we introduce variables. Let $Y_l = e_{j_l}$ and $Z_l = \alpha^{j_l}$. Substituting the variables in the above reveals the shape of this system:

$$\begin{aligned} Y_1 Z_1^{i_1} + \cdots + Y_t Z_t^{i_1} &= s_{i_1} \\ Y_1 Z_1^{i_2} + \cdots + Y_t Z_t^{i_2} &= s_{i_2} \\ &\vdots \\ Y_1 Z_1^{i_r} + \cdots + Y_t Z_t^{i_r} &= s_{i_r} \end{aligned} \tag{8.1}$$

Hence we have a system of equations that is nonlinear with coefficients in $\mathbb{F}_{q^m}$, but linear in the Y_l with coefficients in the field $\mathbb{F}_{q^m}(Z_1, \ldots, Z_t)$. We can try to find the Z_l and then solve for the Y_l by means of Gaussian elimination.

8.3.1 Cooper's method

We will now show how to find the Z_l by means of Gröbner bases and elimination theory. We will first treat the syndromes as constants and in the next section we will discuss an approach due to Cooper treating the syndromes as variables.

We introduce equations specifying which values the variables are allowed to take on.

$$Z_l^n = 1 \text{ for all } 1 \leq l \leq t$$

since α is an nth root of unity. Now, $Y_l = e_{j_l} \in \mathbb{F}_q \setminus \{0\}$ and so we introduce the equations

$$Y_l^q = Y_l \text{ for all } 1 \leq l \leq t$$

Finally, the equations

$$Z_k Z_l p(n, Z_k, Z_l) \text{ for all } 1 \leq k < l \leq t$$

where

$$p(n, X, Y) = \frac{X^n - Y^n}{X - Y} = \sum_{i=0}^{n-1} X^i Y^{n-1-i}$$

are introduced to ensure that Z_k and Z_l are distinct for each k and l, or at least one of them is zero.

Writing $Y = (Y_1, \ldots, Y_t), Z = (Z_1, \ldots, Z_t)$, the previous discussion leads to the ideal defined by the following parametrized set of polynomials in $\mathbb{F}_q[Y, Z]$:

$$Cooper_{q,r,w,Y,Z} = \begin{cases} s_{i_k} - \sum_{l=1}^{w} Y_l Z_l^{i_k} & \text{for all } 1 \leq k \leq r \\ Z_l^n - 1 & \text{for all } 1 \leq l \leq w \\ Y_l^q - Y_l & \text{for all } 1 \leq l \leq w \\ Z_k Z_l p(n, Z_k, Z_l) & \text{for all } 1 \leq k < l \leq w \end{cases}$$

We want to eliminate the variables $Y_1, \ldots, Y_t, Z_2, \ldots, Z_t$, so when computing a Gröbner basis for the ideal generated by the system above we choose the monomial order equal to the lexicographic order with $Y_1 > \cdots > Y_t > Z_t > \cdots > Z_2 > Z_1$. In this case, the elimination ideal $\langle Cooper_{q,r,t,Y,Z} \rangle \cap \mathbb{F}_{q^m}[Z_1]$ will contain a unique polynomial, the roots of which are the error locators we are after.

Proposition 8.3.1. *Suppose that the number of errors t is at most τ. Let $g(Z_1)$ be the monic generator of the ideal $\langle Cooper_{q,r,t,Y,Z} \rangle \cap \mathbb{F}_{q^m}[Z_1]$. Then the zeros of g are the error locators.*

Proof. See [CCS99] chapter 11, proposition 3.6. □

Remark 19. Note that w is set to t in the above!

The following theorem captures all the important ideas underlying the decoding algorithm that we will give afterwards.

Theorem 8.3.2. *Suppose that the number of errors t is at most τ. Let $l(Z_1)$ denote the generator from proposition 8.3.1. Let $g(Z_1)$ be the monic generator of the ideal $\langle Cooper_{q,r,w,Y,Z} \rangle \cap \mathbb{F}_{q^m}[Z_1]$. Then*

$$g(Z_1) = \begin{cases} 1 & \text{if } w < t \\ l(Z_1) & \text{if } w = t \\ Z_1^n - 1 & \text{if } w > t \end{cases}$$

Proof. See [CCS99] chapter 11, theorem 3.7. □

To decode we then do the following: let r be a received word with t errors,

1. Compute the known syndrome vector $s(r)$. If $s(r) = 0$, then no errors have occurred. If not, then go to 2.

2. Compute the reduced Gröbner basis G_w for $w = 1, \ldots$ until $G_w \neq \{1\}$. Let $g(Z_1)$ be the unique element in $G_w \cap \mathbb{F}_{q^m}[Z_1]$. If $\deg(g) > 1$, then too many errors have occurred. Otherwise, find the roots of g. These are the error locators. We know the locators and we also know t, i.e., the number of errors that have occurred and so the error values follow by Gaussian elimination applied to the system 8.1.

Example 8.3.3. *Let C be the ternary Golay code with parameters $[11, 6, 5]$. It can correct at most $\tau = 2$ errors. Its complete defining set is $J(C) = \{1, 3, 4, 5, 9\}$. We will show how to correct two errors in a randomly chosen codeword distorted in two positions by means of an interactive Magma session.*

```
> C := GolayCode(GF(3), false);
> T := [1,3,4,5,9];
> n := Length(C);
> r := Length(C)-Dimension(C);
> t := Floor((MinimumDistance(C)-1)/2);
> q := #Field(C);
> P<x> := PolynomialRing(Field(C));
> F := SplittingField(x^n-1);
> a := RootOfUnity(n, Field(C));
> H := Matrix(F, r, n, [<i,j, a^((j-1)*T[i])> : i in [1..r], j in [1..n]]);
> y := Random(C);
> y;
(1 1 1 2 1 2 2 1 2 2 2)
> y[1] := 2;
> y[4] := 0;
> s := y*Transpose(H);
>
> w := 1;
> Q<[X]> := PolynomialRing(F, 2*w);
> Cooper :=
> [s[i]-&+[X[l]*X[w+(w-l+1)]^(T[i]) : l in [1..w]] : i in [1..r]] cat
> [X[w+(w-l+1)]^(n)-1 : l in [1..w]] cat
> [X[l]^(q)-X[l] : l in [1..w]];
> GroebnerBasis(Cooper);
[
    1
]
>
> w := 2;
> Q<[X]> := PolynomialRing(F, 2*w);
> Cooper :=
```

```
> [s[i]-&+[X[l]*X[w+(w-l+1)]^(T[i]) : l in [1..w]] : i in [1..r]] cat
> [X[w+(w-l+1)]^(n)-1 : l in [1..w]] cat
> [X[l]^(q)-X[l] : l in [1..w]];
> Cooper;
[
    2*X[1]*X[4] + 2*X[2]*X[3] + F.1^211,
    2*X[1]*X[4]^3 + 2*X[2]*X[3]^3 + F.1^149,
    2*X[1]*X[4]^4 + 2*X[2]*X[3]^4 + F.1^151,
    2*X[1]*X[4]^5 + 2*X[2]*X[3]^5 + F.1^131,
    2*X[1]*X[4]^9 + 2*X[2]*X[3]^9 + F.1^205,
    X[4]^11 + 2,
    X[3]^11 + 2,
    X[1]^3 + 2*X[1],
    X[2]^3 + 2*X[2]
]
> GroebnerBasis(Cooper);
[
    X[1] + 2,
    X[2] + 2,
    X[3] + X[4] + F.1^90,
    X[4]^2 + F.1^90*X[4] + F.1^66
]
> R<z> := PolynomialRing(F);
> Roots(UnivariatePolynomial(X[4]^2 + F.1^90*X[4] + F.1^66));
[ <1, 1>, <F.1^66, 1> ]
> a^0;
1
> a^3;
F.1^66
>
```

8.3.2 On- and offline decoding

We introduce the concepts of online and offline decoding. When we are talking about online decoding, we mean that we set up a system of equations every time we receive a word. The solution of this system of equations should solve the decoding problem immediately. When we are talking about offline decoding, we mean that we set up a system of equations in which we treat the syndromes as variables and we do this only once. Solving this system will yield a number of closed formulas in the syndrome variables. When we evaluate these formulas at values specific to a received words, the result should yield a solution to the decoding problem. As we have seen, solving a system of equations is generally a hard problem. The advantage of offline decoding is that we have to solve a system of equations only once. Thereafter, decoding is very simple.

Following this approach, we introduce the variables $S = (S_1, \ldots, S_r)$ for the syndromes

$(s_{i_1}, \ldots, s_{i_r})$. Since the syndromes s_{i_m} are elements of $\mathbb{F}_{q^m}$ we introduce the equations

$$S_i^{q^m} = S_i \text{ for all } 1 \leq i \leq r.$$

The other variables satisfy the same relations as in the previous section. However, observe that the receiver does not know the quantity t, the number of errors that have occurred, in advance. It is convenient to add so-called ghost locations, i.e., $\tau - t$ locations where the Z_l take on the value zero. This amounts to replacing t by τ, the error-correcting capability.

We consider the ideal defined by the following parametrized set of polynomials in $\mathbb{F}_q[S, Y, Z]$

$$Cooper_{q,r,\tau,S,Y,Z} = \begin{cases} S_i - \sum_{l=1}^{\tau} Y_l Z_l^i & \text{for all} \quad 1 \leq i \leq r \\ S_i^{q^m} - S_i & \text{for all} \quad 1 \leq i \leq r \\ Z_l^{n+1} - Z_l & \text{for all} \quad 1 \leq l \leq \tau \\ Y_l^{q-1} - 1 & \text{for all} \quad 1 \leq l \leq \tau \\ Z_l Z_m p(n, Z_l, Z_m) & \text{for all} \quad 1 \leq l < m \leq \tau \end{cases}$$

Now, let G be the reduced Gröbner basis for the ideal defined by the polynomials in $Cooper_{q,r,\tau,S,Y,Z}$ with respect to the lexicographic order induced by

$$Y_t > \cdots > Y_1 > Z_1 > \cdots > Z_t > S_r > \cdots > S_1.$$

It contains a particularly interesting polynomial called the general error locator polynomial.

Theorem 8.3.4. *Every cyclic $[n, k]$-code C with error capacity τ possesses a general error locator polynomial $\mathcal{L}$, i.e., a polynomial in $\mathbb{F}_q[X, T]$ where $X = (X_1, \ldots, X_r)$ satisfying*

- $\mathcal{L}(X, T) = a_0(X) + a_1(X)T + \cdots + a_{\tau-1}(X)T^{\tau-1} + T^{\tau}$ *with* $a_j \in \mathbb{F}[X]$ *for* $0 \leq j \leq \tau - 1$
- *Given a syndrome vector $s = (s_1, \ldots, s_r) \in (\mathbb{F}_{q^m})^r$ corresponding to an error vector of weight $t \leq \tau$ having error locations $j_1, \ldots, j_t$, if we evaluate $\mathcal{L}$ at $X_i = s_i$ for $1 \leq i \leq r$, then the zeros of $\mathcal{L}(s, T)$ are exactly equal to $\alpha^{j_1}, \ldots, \alpha^{j_t}$ and 0 of multiplicity $\tau - t$. In other words, we have the relation*

$$\mathcal{L}(s, T) = T^{\tau-t} \prod_{i=1}^{t} (T - \alpha^{j_i})$$

This polynomial is precisely the element of the reduced Gröbner basis having degree τ with respect to Z_τ.

Proof. See [OS05] theorem 6.8 and theorem 6.9. □

Example 8.3.5. *Let C be the binary cyclic BCH code with parameters $[15, 7, 5]$, i.e., the error-correcting capability $\tau = 2$. This code has $J(C) = \{1, 2, 4, 8, 3, 6, 12, 9\}$ as complete defining set. We want to find the general error locator polynomial of C. We demonstrate this by means of an interactive Magma session.*

```
> n := 15;
> k := 7;
> r := n-k;
```

```
> t := 2;
> T := [1,2,4,8,3,6,12,9];
> P<x> := PolynomialRing(GF(2));
> F := SplittingField(x^n-1);
> a := RootOfUnity(n, GF(2));
> Q<[X]> := PolynomialRing(F, t+r);
> Cooper :=
> [X[t+(r-i+1)]-&+[X[l]^(T[i]) : l in [1..t]] : i in [1..r]] cat
> [X[t+(r-i+1)]^(#F)-X[t+(r-i+1)] : i in [1..r]] cat
> [X[l]^(n+1)-X[l] : l in [1..t]] cat
> [X[l]*X[m]*&+[X[l]^i*X[m]^(n-1-i) : i in [0..n-1]] : l, m in [1..t] | l lt m];
> Cooper;
[
    X[1] + X[2] + X[10],
    X[1]^2 + X[2]^2 + X[9],
    X[1]^4 + X[2]^4 + X[8],
    X[1]^8 + X[2]^8 + X[7],
    X[1]^3 + X[2]^3 + X[6],
    X[1]^6 + X[2]^6 + X[5],
    X[1]^12 + X[2]^12 + X[4],
    X[1]^9 + X[2]^9 + X[3],
    X[10]^16 + X[10],
    X[9]^16 + X[9],
    X[8]^16 + X[8],
    X[7]^16 + X[7],
    X[6]^16 + X[6],
    X[5]^16 + X[5],
    X[4]^16 + X[4],
    X[3]^16 + X[3],
    X[1]^16 + X[1],
    X[2]^16 + X[2],
    X[1]^15*X[2] + X[1]^14*X[2]^2 + X[1]^13*X[2]^3 + X[1]^12*X[2]^4 + X[1]^11*X[2]^5
+ X[1]^10*X[2]^6 +
        X[1]^9*X[2]^7 + X[1]^8*X[2]^8 + X[1]^7*X[2]^9 + X[1]^6*X[2]^10 + X[1]^5*X[2]^11
+ X[1]^4*X[2]^12 +
        X[1]^3*X[2]^13 + X[1]^2*X[2]^14 + X[1]*X[2]^15
]
> GroebnerBasis(Cooper);
[
    X[1] + X[2] + X[10],
    X[2]^2 + X[2]*X[10] + X[6]*X[10]^14 + X[10]^2,
    X[2]*X[10]^15 + X[2],
    X[3] + X[6]^4*X[10]^12 + X[6]^2*X[10]^3 + X[6]*X[10]^6,
    X[4] + X[6]^4,
    X[5] + X[6]^2,
    X[6]^8 + X[6]^4*X[10]^12 + X[6]^2*X[10]^3 + X[6]*X[10]^6,
    X[6]*X[10]^15 + X[6],
```

```
    X[7] + X[10]^8,
    X[8] + X[10]^4,
    X[9] + X[10]^2,
    X[10]^16 + X[10]
]
```

Theorem 8.3.4 ensures that the reduced Gröbner basis contains the unique general error locator polynomial. It should be of degree 2 with respect to Z_2. Indeed, we find $L(X_1, X_2, T) = T^2 + TX_1 + X_4X_1^14 + X_1^2$.

8.3.3 Newton identities based method

In this section we will only consider binary codes. We call the polynomial in theorem 8.3.4 the generalized error locator polynomial because it generalizes the following notion:

Definition 8.3.6 (Plain error locator polynomial)**.**

$$\sigma(Z) = \prod_{k=1}^{t}(1 - \alpha^{j_k}Z)$$

The plain error locator polynomial encodes the inverse of the error locators as its roots and plays an important role in many of the traditional BCH decoding algorithms. As we have observed before, in order to solve 8.1 it suffices to find the error locators. This is equivalent to finding the plain error locator polynomial, as the error locators will follow from the roots. How do we find $\sigma(Z)$? Let us expand $\sigma(Z)$ to find

$$\sigma(Z) = \sigma_t Z^t + \sigma_{t-1}Z^{t-1} + \cdots + \sigma_1 Z + 1$$

where the σ_i are the elementary symmetric functions in the error locators $Z_1, \ldots, Z_t$.

$$\sigma_i = (-1)^i \sum_{1 \le j_1 < j_2 < \cdots < j_i \le t} Z_{j_1} Z_{j_2} \ldots Z_{j_i}, \text{ for } 1 \le i \le t$$

It follows that finding the plain error locator polynomial is equivalent to finding σ_i for all $1 \le i \le t$. Now, the syndromes satisfy the so-called generalized Newton identities (see [MS77] theorem 24)

$$s_{i+t} + \sum_{j=1}^{t} \sigma_j s_{i+t-j} = 0 \text{ for } 1 \le i \le t$$

In the binary case this is equivalent with

$$\begin{cases} s_i + \sum_{j=1}^{i-1} \sigma_j s_{i-j} + i\sigma_i = 0 & \text{for} \quad 1 \le i \le t \\ s_i + \sum_{j=1}^{t} \sigma_j s_{i-j} = 0 & \text{for} \quad 1 + t \le i \le n + t \end{cases}$$

We will use these equations to set up a system of equations. First, we will consider the syndromes as variables, i.e., we will focus on offline decoding. Clearly, we have

$$s_{i+n} = e(a^{i+n}) = e(a^i) = s_i \text{ for all } 1 \le i \le n$$

and

$$s_i^2 = (e(a^i))^2 = e(a^{2i}) = s_{2i} \text{ for all } 1 \le i \le n$$

Now, $\alpha \in \mathbb{F}_{2^m} \setminus \{0\}$ and so $Z_l \in \mathbb{F}_{2^m} \setminus \{0\}$. Therefore, we find the relation

$$\sigma_i^{2^m} = \sigma_i \text{ for all } 1 \leq i \leq t$$

Next, we introduce the variables $S = (S_1, \ldots, S_{n+w})$ and $X = (X_1, \ldots, X_w)$ for the syndromes and σ_i respectively. Finally, we consider the ideal generated by the following polynomials in $\mathbb{F}_2[S, X]$:

$$Newton_{w,n,S,X} = \begin{cases} X_i^{2^m} - X_i & \text{for all } 1 \leq i \leq w \\ S_i^{2^m} - S_i & \text{for all } 1 \leq i \leq n \\ S_i^2 - S_{2i} & \text{for all } 1 \leq i \leq n \\ S_{i+n} - S_i & \text{for all } 1 \leq i \leq w \\ S_i + \sum_{j=1}^{i-1} X_j S_{i-j} + iX_i & \text{for all } 1 \leq i \leq w \\ S_i + \sum_{j=1}^{w} X_j S_{i-j} & \text{for all } 1 + w \leq i \leq n + w \end{cases}$$

In the case of online decoding, we add to $Newton_{w,n,S,X}$ the polynomials $S_i - s_i$ for $i \in J(C)$ and $X_{t+1}, \ldots, X_w$.

Proposition 8.3.7. *Let I be the ideal generated by the polynomials in $Newton_{w,n,S}$ and polynomials $S_i - s_{i_k}$ for $1 \leq k \leq r$ and $\sigma_{t+1}, \ldots, \sigma_w$. Then*

$$I = \langle X_1 - \sigma_t, \ldots, X_t - \sigma_t, X_{t+1}, \ldots, X_w, S_i - s_i, i \in \{0, \ldots, n-1\} \setminus J(C) \rangle$$

Proof. See [ABF09] theorem 10. □

To decode we then do the following: let r be a received word with t errors,

1. Compute the known syndrome vector $s(r)$. If $s(r) = 0$, then no errors have occurred. If not, then go to 2.

2. Compute the reduced Gröbner basis G_w for $w = 1, \ldots$ until $G_w \neq \{1\}$. Then $w = t$ and G_w contains the polynomials $X_1 - \sigma_t, \ldots, X_t - \sigma_t$.

To summarize: the resulting value of w will yield t and the solution will yield the σ_i. We then know the plain error locator polynomial, the roots of which are easy to find. From the error locators the error locations are readily found. Since we are working over $\mathbb{F}_2$ the error values are all equal to 1, so the problem is solved.

Example 8.3.8. *Let C be the binary BCH code with parameters $[31, 16, 7]$ and complete defining set $J(C) = \{1, 2, 4, 8, 16, 5, 10, 20, 9, 18, 7, 14, 28, 25, 19\}$. Since $J(C)$ contains 4 consecutive integers the BCH bound says that $d(C) \geq 5$, hence most BCH decoding algorithms are only able to correct at most two errors. However, the true error-correcting capability of C is $\tau = 3$. We will now try to correct three errors. Let*

$$r = (0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1)$$

and assume for now that we do not know the number of errors in r, but we do know that it is at most τ. The known syndromes are $s_1 = a^5, s_5 = a^8, s_7 = a^{26}$. We illustrate the decoding process by means of an interactive Magma session.

```
> n := 31;
> tau := 3;
> P<x> := PolynomialRing(GF(2));
> F := SplittingField(x^n-1);
> a := RootOfUnity(n, GF(2));
>
> w := 1;
> Q<[X]> := PolynomialRing(F, 2*n+tau, "grevlex");
> Newton :=
> [X[2*n+i]^#F-X[2*n+i] : i in [1..w]] cat
> [X[i]^2-X[2*i] : i in [1..n]] cat
> [X[i+n]-X[i] : i in [1..w]] cat
> [X[1] + X[2*n+1]] cat
> [X[i] + &+[X[2*n+j]*X[i-j] : j in [1..i-1]] + i*X[2*n+i]: i in [2..w]] cat
> [X[i] + &+[X[2*n+j]*X[i-j] : j in [1..w]]: i in [1+w..n+w]] cat
> [X[2*n+i] : i in [w+1..tau]] cat
> [X[1]-a^5, X[5]-a^8, X[7]-a^26];
>
> GroebnerBasis(Newton);
[
    1
]
> w := 2;
> Q<[X]> := PolynomialRing(F, 2*n+tau, "grevlex");
> Newton :=
> [X[2*n+i]^#F-X[2*n+i] : i in [1..w]] cat
> [X[i]^2-X[2*i] : i in [1..n]] cat
> [X[i+n]-X[i] : i in [1..w]] cat
> [X[1] + X[2*n+1]] cat
> [X[i] + &+[X[2*n+j]*X[i-j] : j in [1..i-1]] + i*X[2*n+i]: i in [2..w]] cat
> [X[i] + &+[X[2*n+j]*X[i-j] : j in [1..w]]: i in [1+w..n+w]] cat
> [X[2*n+i] : i in [w+1..tau]] cat
> [X[1]-a^5, X[5]-a^8, X[7]-a^26];
> GroebnerBasis(Newton);
[
    1
]
> w := 3;
> Q<[X]> := PolynomialRing(F, 2*n+tau, "grevlex");
> Newton :=
> [X[2*n+i]^#F-X[2*n+i] : i in [1..w]] cat
> [X[i]^2-X[2*i] : i in [1..n]] cat
> [X[i+n]-X[i] : i in [1..w]] cat
> [X[1] + X[2*n+1]] cat
> [X[i] + &+[X[2*n+j]*X[i-j] : j in [1..i-1]] + i*X[2*n+i]: i in [2..w]] cat
> [X[i] + &+[X[2*n+j]*X[i-j] : j in [1..w]]: i in [1+w..n+w]] cat
> [X[2*n+i] : i in [w+1..tau]] cat
```

```
> [X[1]-a^5, X[5]-a^8, X[7]-a^26];
> GroebnerBasis(Newton);
[
    X[1] + F.1^5,
    X[2] + F.1^10,
    X[3] + F.1^24,
    X[4] + F.1^20,
    X[5] + F.1^8,
    X[6] + F.1^17,
    X[7] + F.1^26,
    X[8] + F.1^9,
    X[9] + F.1^2,
    X[10] + F.1^16,
    X[11] + F.1^4,
    X[12] + F.1^3,
    X[13] + F.1^16,
    X[14] + F.1^21,
    X[15] + F.1^16,
    X[16] + F.1^18,
    X[17] + F.1^12,
    X[18] + F.1^4,
    X[19] + F.1^13,
    X[20] + F.1,
    X[21] + F.1^2,
    X[22] + F.1^8,
    X[23] + F.1^8,
    X[24] + F.1^6,
    X[25] + F.1^22,
    X[26] + F.1,
    X[27] + F.1^4,
    X[28] + F.1^11,
    X[29] + F.1^2,
    X[30] + F.1,
    X[31] + 1,
    X[32] + F.1^5,
    X[33] + F.1^10,
    X[34] + F.1^24,
    X[36] + F.1^8,
    X[38] + F.1^26,
    X[40] + F.1^2,
    X[42] + F.1^4,
    X[44] + F.1^16,
    X[46] + F.1^16,
    X[48] + F.1^12,
    X[50] + F.1^13,
    X[52] + F.1^2,
    X[54] + F.1^8,
```

```
    X[56] + F.1^22,
    X[58] + F.1^4,
    X[60] + F.1^2,
    X[62] + 1,
    X[63] + F.1^5,
    X[64] + F.1^5,
    X[65] + F.1^4
]
```

So we find that $\sigma_1 = a^5, \sigma_2 = a^5$, *and* $\sigma_3 = a^4$. *Plugging these into* $\sigma(Z)$ *and finding its roots yields:*

```
> Q<z> := PolynomialRing(F);
> Roots(F.1^4*z^3+F.1^5*z^2+F.1^5*z+1);
[ <F.1^6, 1>, <F.1^24, 1>, <F.1^28, 1> ]
```

It follows that the inverted error locators are a^6, a^{24}, *and* a^{28} *and therefore the error locators are* a^{25}, a^7, *and* a^3. *Finally, the locators yield the error positions* $3, 5$, *and* 25.

Chapter 9

Decoding general linear codes

9.1 The method of unknown syndromes

In this section, we will forget about any extra structure and focus on general linear codes. We will set up a system of quadratic equations, the solution to which will solve the decoding problem.
Throughout this section, let C be a linear $[n, k, d]$-code over $\mathbb{F}_q$ having parity check matrix H. Let $b_1, \ldots, b_n$ be a basis for the vector space $\mathbb{F}_q^n$. Write B for the $n \times n$ matrix having rows equal to $b_1, \ldots, b_n$. If we let h_i denote the ith row of H, then we can express this in the basis as

$$h_i = \sum_{j=1}^{n} a_{ij} b_j$$

for suitable scalars $a_{i1}, \ldots, a_{in} \in \mathbb{F}_q$. It follows that there exists an $(n-k) \times n$ matrix A such that $H = AB$ where $A = (a_{ij})$.

Definition 9.1.1 (Unknown syndrome)**.** *The unknown syndrome of a word e with respect to B is $u(e) = Be^t$.*

Remark 20. The unknown syndrome thus defined is a generalized notion than that of the unknown syndrome in the context of cyclic codes.

Proposition 9.1.2. *The matrix B is invertible.*

Proof. The row space of B equals $\mathbb{F}_q^n$. Hence, the rank of B is n and the result follows. □

The receiver receives a word $y = c + e$ composed of a sent codeword $c \in C$ and an error vector $e \in \mathbb{F}_q^n$. Recall that the problem we are faced with is the recovery of the error vector e. However, since the matrix B is invertible, we can simply recover e by computing $B^{-1}u(e) = B^{-1}Be^t = e^t$. Thus, our new goal is trying to find the unknown syndrome vector $u(e)$.

Definition 9.1.3 (Star product)**.** *The star product between $x, y \in \mathbb{F}_q^n$ is defined as*

$$x * y = (x_1 y_1, \ldots, x_n y_n)$$

Definition 9.1.4 (Structure constants)**.** *For every $1 \leq i, j \leq n$ the star product $b_i * b_j$ can be expressed in the basis vectors as*

$$b_i * b_j = \sum_{l=1}^{n} \mu_l^{ij} b_l$$

The scalars μ_l^{ij} are called the structure constants of the basis vectors $b_1, \ldots, b_n$.

Definition 9.1.5 (Matrix of unknown syndromes)**.** *The $n \times n$ matrix of unknown syndromes of a word y, denoted $U(e)$, is given by*

$$u_{ij}(e) = (b_i * b_j) \cdot e = \sum_{l=1}^{n} \mu_l^{ij} u_l(e)$$

where $u_l(e)$ is the lth entry of the vector of unknown syndromes, $u(e)$.

Proposition 9.1.6. *We have that $U(e) = B \operatorname{diag}(e) B^T$. Furthermore, the rank of $U(e)$ is equal to $\operatorname{wt}(e)$.*

Proof. $u_{ij}(e) = (b_i * b_j) \cdot e = (b_{i1}b_{j1}, \ldots, b_{in}b_{jn}) \cdot e = \sum_{l=1}^{n} b_{il} e_l b_{jl}$. It follows that $U(e) = B \operatorname{diag}(e) B^T$. Now, B is invertible, hence $\operatorname{rk} B = \operatorname{rk} B^T = n$. Therefore,

$$\operatorname{rk} U(e) = \operatorname{rk} B \operatorname{diag}(e) B^T = \operatorname{rk} \operatorname{diag}(e) B^T = \operatorname{rk} \operatorname{diag}(e) = \operatorname{wt}(e)$$

. □

Definition 9.1.7 (MDS basis, matrix)**.** *Let $b_1, \ldots, b_n$ be a basis for $\mathbb{F}_q^n$. Write B_i for the matrix having rows equal to $b_1, \ldots, b_i$. If all $i \times i$ submatrices of B_i have full rank, for $i = 1, \ldots, n$, then we call $b_1, \ldots, b_n$ an ordered MDS basis and the associated matrix B an MDS matrix.*

The name MSD basis is derived from the fact that the code described by parity check matrix B_i is an MDS code for every $1 \leq i \leq n$.

Definition 9.1.8 (Vandermonde basis, matrix)**.** *Assume that $n \leq q$. Let $x = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ be a vector of pairwise distinct elements, and let $b_i = (x_1^{i-1}, \ldots, x_n^{i-1})$. We call $b_1, \ldots, b_n$ a Vandermonde basis and the associated matrix B a Vandermonde matrix.*

Proposition 9.1.9. *If $b_1, \ldots, b_n \in \mathbb{F}_q^n$ is a Vandermonde basis, then it is an MDS basis.*

If there exists an element of order n, say α, we may pick $x_j = \alpha^{j-1}$, $j = 1, \ldots, n$ to obtain a Vandermonde basis.
There exists an integer m such that $n \leq q^m$. Hence, after a finite field extension, we may assume that our field satisfies the assumptions in definition 9.1.7. In the following, the finite field under consideration will be $\mathbb{F}_{q^m}$.

Definition 9.1.10. *Let $M = (m_{ij})$ be an $m \times n$ matrix over some field. We define M_{vw} to be the $v \times v$ submatrix of M given by the first v rows of M and then restricting to the first w positions, i.e.,*

$$M_{vw} = (m_{ij})_{1 \leq i \leq v, 1 \leq j \leq w}$$

We will also write M_v for M_{vn}.

Proposition 9.1.11. *Assume that B is an MDS matrix. Write $t = \operatorname{wt}(e)$. Then*

$$\operatorname{rk} U_{nv}(e) = \min\{v, t\}$$

Proof. From the identity $U_{nv}(e) = B\,\mathrm{diag}(e)B_v^T$ it follows that

$$\mathrm{rk}\,U_{nv}(e) \leq \min\{\mathrm{rk}\,B, \mathrm{rk}\,\mathrm{diag}(e), \mathrm{rk}\,B_v^T\} = \min\{n, t, v\} = \min\{t, v\}.$$

We will show that $\mathrm{rk}\,U_{nv}(e) \geq \min\{t, v\}$. After a permutation of the columns of B, we may assume that the t nonzero entries of e, denoted $e_1, \ldots, e_t$, are on the first t positions. B remains an MDS matrix, as the rank is permutation invariant. Write $e' = (e_1, \ldots, e_t)$. Then $B\,\mathrm{diag}(e)B_v^T$ has $B_t\,\mathrm{diag}(e')B_{vt}^T$ as a submatrix. Now, $\mathrm{diag}(e')$ is invertible, since its diagonal consists of only nonzero entries. It follows that $\mathrm{rk}\,B_t\,\mathrm{diag}(e')B_{vt}^T = \mathrm{rk}\,B_tB_{vt}^T$. Now, B_t is a submatrix of B, an MDS matrix, so it has full rank and is invertible. It follows that $\mathrm{rk}\,B^tB_{vt}^T = \mathrm{rk}\,B_{vt}^T$, but this equals $\min\{v, t\}$ as B is MDS. Hence, since $U_{nv}(e)$ contains a submatrix of rank $\min\{v, t\}$ it follows that $\mathrm{rk}\,U_{nv}(e) \geq \min\{v, t\}$. □

By proposition 9.1.6, the rank of $U(e)$ is equal to $\mathrm{wt}(e)$. Write $t = \mathrm{wt}(e)$. It follows that any $t + 1$ vectors in the row space of $U(e)$ are linearly dependent. In particular, there exist $v_1(e), \ldots, v_t(e) \in \mathbb{F}_q$ such that $\sum_{j=1}^t u_{ij}(e)v_j(e) = u_{i(t+1)}$ for all $i = 1, \ldots, n$. By substitution of, it follows that this equals $\sum_{j=1}^t \left(\sum_{l=1}^n \mu_l^{ij} u_l(e)\right) v_j(e) = \sum_{l=1}^n \mu_l^{i(t+1)} u_l(e)$.

Definition 9.1.12. *Define the linear functions U_{ij} in the variables $U_1, \ldots, U_n$ by*

$$U_{ij} = \sum_{l=1}^{n} \mu_l^{ij} U_l$$

Evaluating $(U_1, \ldots, U_n)$ at $u(e) = (u_1, \ldots, u_n)$ gives $U_{ij} = u_{ij}$.

Definition 9.1.13. *The ideal $I(t, U, V)$ is the ideal in $\mathbb{F}_q[U_1, \ldots, U_n, V_1 \ldots, V_t]$ generated by the elements $\sum_{j=1}^t U_{ij}V_j - U_{i(t+1)}$ for $i = 1, \ldots, n$.*

Using the relation $H = AB$ we can express the known syndrome of y in terms of the unknown syndrome of y as follows:

$$s_i(y) = s_i(e) = h_i \cdot e = (\sum_{j=1}^{n} a_{ij}b_j) \cdot e = \sum_{j=1}^{n} a_{ij}(b_j \cdot e) = \sum_{j=1}^{n} a_{ij}u_j(e) \tag{9.1}$$

Definition 9.1.14. *The ideal $J(y)$ is the ideal in $\mathbb{F}_q[U_1, \ldots, U_n]$ generated by the elements $\sum_{l=1}^n a_{jl}U_l - s_j(y)$ for $j = 1, \ldots, n - k$.*

Definition 9.1.15. *The ideal $J(t, y)$ is the ideal in $\mathbb{F}_q[U_1, \ldots, U_n, V_1 \ldots, V_t]$ generated by $I(t, U, V)$ and $J(y)$.*

Lemma 9.1.16. *If $r = c + e$ with $c \in C$ and $e \in \mathbb{F}_q^n$ with $\mathrm{wt}(e) = t$, then there exists a v such that $(u(e), v)$ is a zero of $J(t, r)$.*

Proof. Clearly, $u(e)$ is a solution of $J(r)$ by 9.1. By proposition $\mathrm{rk}\,U_{nv}(e) = \min\{v, t\}$. Hence, if $v > t$, then $\mathrm{rk}\,U_{nv}(e) = t$. It follows that the $(t + 1)$th column of $U_{n(t+1)}(e)$ is a linear combination of the first t columns. Therefore, there exists a v solving $J(t, r)$. □

Lemma 9.1.17. *Let (u, v) be a solution of $J(t, r)$. Then there exists a unique e with $\mathrm{wt}(e) \leq t$ such that $u = u(e)$. Moreover, $r = c + e$ for some $c \in \mathbb{F}_{q^m}C$ for some $m \in \mathbb{Z}_{>0}$.*

Proof. We have that (u, v) is a solution of $J(t, r)$, so in particular u is a solution of $J(r)$. Hence, it is the vector of unknown syndromes with respect to a unique e, i.e., $u = u(e)$. Moreover, it follows that the $(t+1)$th column of $U(e)$ is a linear combination of the first t columns. Therefore, $\operatorname{rk} U_{n(t+1)}(e) \leq t$. On the other hand, proposition 9.1.11 says that $\operatorname{rk} U_{n(t+1)}(e) = \min\{t+1, \operatorname{wt}(e)\}$. Hence, $\min\{t+1, \operatorname{wt}(e)\} \leq t$ implies that $\operatorname{wt}(e) \leq t$. Now, $s_i(r) = \sum_{i=1}^{n} a_{ij} u_j = s_i(e) \in \mathbb{F}_{q^m}$ for $1 \leq i \leq n$. It follows that $s(e) = s(r)$. Hence the syndrome of $r - e$ equals 0, but this implies that $r - e \in \mathbb{F}_{q^m} C$. Hence, there exists a $c \in \mathbb{F}_{q^m} C$ such that $r = c + e$. □

Lemma 9.1.18. *If (u, v) and (u, w) are distinct solutions of $J(t, r)$, then there exists a solution (u, z) of $J(t', r)$ for some $t' < t$. Moreover, if $t' = 0$, then $r \in \mathbb{F}_{q^m} C$ for some $m \in \mathbb{Z}_{>0}$.*

Proof. Let (u, v) and (u, w) be solutions of $J(t, r)$ such that $v \neq w$. Lemma 9.1.17 says that $u = u(e)$. Moreover, (u, v) and (u, w) are solutions of $I(t, U, V)$ and thus we have

$$\sum_{l=1}^{t} u_{il} v_l = u_{i(t+1)} \text{ and } \sum_{l=1}^{t} u_{il} w_l = u_{i(t+1)} \text{ for } 1 \leq i \leq n$$

It follows that

$$\sum_{l=1}^{t} u_{il}(v_l - w_l) = 0 \text{ for } 1 \leq i \leq n$$

By assumption, $v \neq w$, so there exists at least one l such $v_l - w_l \neq 0$. Hence the first t columns of U are linearly dependent, i.e., there exists a $t' < t$ such that the $t'+1$th column of U is a linear combination of the first t'. It follows that there exists a solution (u, z) of $J(t', r)$. Now, suppose that $t' = 0$. Then lemma 9.1.17 says that $u = u(e)$ for a unique e with $\operatorname{wt}(e) = 0$ and $r = c + e$ for some $c \in \mathbb{F}_{q^m} C$. Since $\operatorname{wt}(e) = 0$, we see that $e = 0$ and so $r = c \in \mathbb{F}_{q^m} C$. □

Theorem 9.1.19. *Let B be an MDS matrix whose rows have structure constants μ_l^{ij} associated with it. Let H be a parity check matrix of a code C such that $H = AB$. Let $r = c + e$ be a received word composed of a codeword $c \in C$ and error vector $e \in \mathbb{F}_q^n$. Suppose that $0 < \operatorname{wt}(e) \leq \lfloor \frac{d-1}{2} \rfloor$. Let t be the smallest positive integer such that there exists a solution (u, v) of $J(t, r)$ over $\overline{\mathbb{F}}_q$. Then $\operatorname{wt}(e) = t$ and the solution is unique and satisfies $u = u(e)$.*

Proof. Observe that lemma 9.1.16 says that there exists a v such that $(u(e), v)$ is a solution of $J(\operatorname{wt}(e), r)$. Now, $J(s, r)$ is parametrized by s and our observation shows that for at least one such s the system is solvable. Now, let t be minimal with this property. In particular, we know that $t \leq \operatorname{wt}(e)$. Let $(\overline{u}, \overline{v})$ be an arbitrary corresponding solution to $J(t, r)$. First, we prove that $\overline{u}$ is unique. Lemma 9.1.17 says that there exists a unique $\overline{e}$ with $\operatorname{wt} \overline{e} \leq t$ with $\overline{u} = \overline{u}(\overline{e})$ and $r = c + \overline{e}$ for some $c \in \mathbb{F}_{q^m} C$. It follows that $\operatorname{wt} \overline{e} \leq t \leq \operatorname{wt} e$. Hence,

$$\operatorname{wt} \overline{e} - e = d(\overline{e}, e) \leq d(\overline{e}, 0) + d(0, e) = \operatorname{wt} \overline{e} + \operatorname{wt} e \leq 2 \operatorname{wt} e \leq d - 1 < d(C) = \operatorname{wt}(C)$$

but then we have found a codeword of weight less than $\operatorname{wt}(C)$. This means that it is the zero codeword, i.e., $\overline{e} = e$. Hence $\overline{u} = \overline{u}(\overline{e}) = \overline{u}(e)$, and by uniqueness this equals $u(e)$. Since $\overline{u}$ was picked arbitrarily, it follows that there is in fact a unique $\overline{u}$ equal to $u(e)$.
Next, we prove that $\overline{v}$ is unique. To this end, let $(u, \widetilde{v})$ be a solution of $J(t, r)$ such that $\overline{v} \neq \widetilde{v}$. Observe that r is not a codeword, since $\operatorname{wt}(e) > 0$. Hence, lemma 9.1.18 says that there exists a solution (u, z) of $J(t', r)$ for some $t' < t$. However, this contradicts the minimality of t. Hence, $\overline{v}$ is unique. Since both $\overline{u}$ and $\overline{v}$ are unique, the solution itself is unique. □

Theorem 9.1.19 shows that $u = u(e)$. Since $e \in \mathbb{F}_q$ it follows that $u \in \mathbb{F}_q$. From this, using the equations, we can deduce that $v \in \mathbb{F}_q$ as well. Hence, the entire solution lies in $\mathbb{F}_q$. It follows that $V(\langle J(t,r)\rangle) = V(\langle J(t,r)\rangle + \langle U_1^q - 1, \ldots, U_n^q - 1, V_1^q - 1, \ldots, V_t^q - 1\rangle)$. Moreover, it follows from Seidenberg's theorem 2.5.21 that $\langle J(t,r)\rangle + \langle U_1^q - 1, \ldots, U_n^q - 1, V_1^q - 1, \ldots, V_t^q - 1\rangle$ is radical.

Theorem 9.1.20 (Lexicographic case)**.** *Let $r = c+e$ be a received word composed of codeword $c \in C$ and error vector $e \in \mathbb{F}_q^n$. Assume that $\mathrm{wt}(e) \leq \lfloor \frac{d(C)-1}{2} \rfloor$. Let t be the smallest integer such that $V(J(t,r) + \langle x_1^{q^m} - 1, \ldots, x_n^{q^m} - 1\rangle) \neq \emptyset$. Then this solution is unique. Moreover, the reduced Gröbner basis for the ideal $J(t,r)$ with respect to the lexicographic order has the shape*

$$\begin{aligned} U_i - u_i(e), i = 1, \ldots, n \\ V_j - v_j, j = 1, \ldots, t \end{aligned}$$

Proof. By adding the field equations, lemma 2.5.21 tells us that the ideal is radical. Moreover, by theorem 9.1.19 the ideal is zero-dimensional. Hence lemma 2.5.25 says that the reduced Gröbner basis looks like the claim. □

Theorem 9.1.21 (General case)**.** *Let $r = c + e$ be a received word composed of codeword $c \in C$ and error vector $e \in \mathbb{F}_q^n$. Assume that $\mathrm{wt}(e) \leq \lfloor \frac{d(C)-1}{2} \rfloor$. Let t be the smallest integer such that $V(J(t,r)) \neq \emptyset$. Then this solution is unique. Moreover, the reduced Gröbner basis for the ideal $J(t,r)$ with respect to any monomial order has the shape*

$$\begin{aligned} U_i - u_i(e), i = 1, \ldots, n \\ V_j - v_j, j = 1, \ldots, t \end{aligned}$$

Proof. See [BP09] theorem 51. □

Example 9.1.22. *Consider the following example where we let C be the Golay code over $\mathbb{F}_3$ and introduce two errors to a random codeword. We present the example as an interactive Magma session.*

```
> C := GolayCode(GF(3), false);
> C;
[11, 6, 5] "Unextended Golay Code" Linear Code over GF(3)
Generator matrix:
[1 0 0 0 0 0 2 0 1 2 1]
[0 1 0 0 0 0 1 2 2 2 1]
[0 0 1 0 0 0 1 1 1 0 1]
[0 0 0 1 0 0 1 1 0 2 2]
[0 0 0 0 1 0 2 1 2 2 0]
[0 0 0 0 0 1 0 2 1 2 2]
> r := Random(C);
> r;
(0 0 1 2 1 0 2 1 0 0 2)
> r[3] := 0; r[11] := 1;
> t := 2;
```

```
> FE, _ := ext<Field(C) | Ceiling(Log(#Field(C), Length(C)))>;
> C, _ := ExtendFieldCode(C, FE);
> Field(C);
Finite field of size 3^3
> a := [PrimitiveElement(Field(C))^(i-1) : i in [1..Length(C)]];
> B := Matrix(Field(C), Length(C), Length(C), [<i, j, a[j]^(i-1)> : i, j in [1..Length(C)]
> mu := [[StarProduct(B[i], B[j])*B^(-1) : j in [1..Length(C)]] : i in [1..Length(C)]];
> A := ParityCheckMatrix(C)*B^(-1);
> s := r*Transpose(ParityCheckMatrix(C));
> P<[X]> := PolynomialRing(Field(C), Length(C)+t, "grevlex");
> J := [&+[A[j, l]*X[l] : l in [1..Length(C)]] - s[j] : j in [1..Length(C)-Dimension(C)]];
> I := [&+[&+[mu[i,j,l]*X[l] : l in [1..Length(C)]]*X[Length(C)+j] : j in [1..t]]-&+[mu[i,
: l in [1..Length(C)]] : i in [1..Length(C)]];
> sys := J cat I;
> sys;
[
    FE.1^23*X[1] + FE.1^58*X[2] + FE.1^5*X[3] + FE.1^68*X[4] + FE.1^69*X[5] + FE.1^12*X[6]
+ FE.1^57*X[7] + FE.1^44*X[8] + FE.1^49*X[9] +
        FE.1^69*X[10] + FE.1^44*X[11],
    FE.1^7*X[1] + FE.1^56*X[2] + FE.1^51*X[3] + FE.1^50*X[4] + FE.1^23*X[5] + FE.1^50*X[6]
+ FE.1^22*X[7] + FE.1^71*X[8] + FE.1^17*X[9] +
        FE.1^28*X[10] + FE.1^66*X[11] + 1,
    FE.1^42*X[1] + FE.1^76*X[2] + FE.1^49*X[3] + FE.1^36*X[4] + FE.1^16*X[5] +
FE.1^15*X[6] + 2*X[7] + FE.1^17*X[8] + FE.1^53*X[9] +
        FE.1^55*X[10] + FE.1^30*X[11],
    FE.1^49*X[1] + FE.1^8*X[2] + FE.1^17*X[3] + FE.1^22*X[4] + FE.1^60*X[5] + FE.1^38*X[6]
+ FE.1^65*X[7] + FE.1^3*X[8] + FE.1^76*X[9] +
        FE.1^4*X[10] + FE.1^45*X[11] + 1,
    FE.1^58*X[1] + FE.1^48*X[2] + FE.1^68*X[3] + FE.1^14*X[4] + FE.1^52*X[5] +
2*X[6] + FE.1^36*X[7] + FE.1^25*X[9] + FE.1^3*X[10] +
        FE.1^4*X[11] + 1,
    X[1]*X[12] + X[2]*X[13] + 2*X[3],
    X[2]*X[12] + X[3]*X[13] + 2*X[4],
    X[3]*X[12] + X[4]*X[13] + 2*X[5],
    X[4]*X[12] + X[5]*X[13] + 2*X[6],
    X[5]*X[12] + X[6]*X[13] + 2*X[7],
    X[6]*X[12] + X[7]*X[13] + 2*X[8],
    X[7]*X[12] + X[8]*X[13] + 2*X[9],
    X[8]*X[12] + X[9]*X[13] + 2*X[10],
    X[9]*X[12] + X[10]*X[13] + 2*X[11],
    X[10]*X[12] + X[11]*X[13] + FE.1^15*X[1] + FE.1^53*X[2] + FE.1^49*X[3] + FE.1^63*X[4]
+ FE.1^7*X[5] + FE.1^6*X[6] + FE.1^41*X[7] +
        FE.1^32*X[8] + FE.1^78*X[9] + FE.1^54*X[10] + FE.1^48*X[11],
    X[11]*X[12] + FE.1^55*X[1]*X[13] + FE.1^13*X[2]*X[13] + FE.1^9*X[3]*X[13] +
FE.1^23*X[4]*X[13] + FE.1^47*X[5]*X[13] +
        FE.1^46*X[6]*X[13] + FE.1*X[7]*X[13] + FE.1^72*X[8]*X[13] + FE.1^38*X[9]*X[13]
+ FE.1^14*X[10]*X[13] + FE.1^8*X[11]*X[13] +
```

```
        FE.1^23*X[1] + FE.1^50*X[2] + FE.1^44*X[3] + FE.1^16*X[4] + FE.1^49*X[5]
+ FE.1^69*X[6] + FE.1^37*X[7] + FE.1^68*X[8] +
        FE.1^78*X[9] + FE.1^20*X[10] + FE.1^32*X[11]
]
> GroebnerBasis(sys);
[
    X[1] + 2,
    X[2] + FE.1^31,
    X[3] + FE.1^42,
    X[4] + FE.1^13,
    X[5] + FE.1^74,
    X[6],
    X[7] + FE.1^46,
    X[8] + FE.1^77,
    X[9] + FE.1^38,
    X[10] + FE.1^39,
    X[11] + FE.1^60,
    X[12] + FE.1^12,
    X[13] + FE.1^71
]
```

These results readily lead to the algorithm 17 for decoding a general code.

input : An $(n-k) \times n$ parity check matrix H describing an $[n,k,d]$-code C, a word $y = c + e \in \mathbb{F}_q^n$ composed of a codeword $c \in C$ and an error vector $e \in \mathbb{F}_q^n$, and a monomial ordering $>$.
output: The codeword c.
begin
  $G := [1]$;
  $t := 1$;
  while $G = [1]$ **do**
    $G :=$ `ReducedGroebnerBasis(`$J(t,y), >$`)` ;
    $t := t + 1$;
  end
  G is of the form $[U_j - u_j, V_l - v_l]_{1 \le j \le n, 1 \le l \le t}$
  $u :=$ [`-ConstantCoefficient(`$G[1]$`)`, ..., `-ConstantCoefficient(`$G[n]$`)`] ;
  $e := B^{-1}u$;
  return $y - e$
end

Algorithm 17: General decoding algorithm

9.2 Complexity

By performing Gaussian elimination on the $n-k$ linear polynomials it is possible to express the first $n-k$ variables, i.e., $U_1, \ldots, U_{n-k}$, in terms of the other k and then substitute them into the quadratic part to obtain a system of n quadratic polynomials in $k+t$ variables. If we embed the ring $\mathbb{F}_q[U'_1, \ldots, U'_k, V_1, \ldots, V_t]$ in the ring $\mathbb{F}_q[U'_1, \ldots, U'_n, V_1, \ldots, V_t]$ we get a system that is affine bilinear and has at most as many equations as variables, so it seems possible (see corollary 3 in [FSEDS11]) to use the bound in theorem 4.11.10 to bound the complexity by

$$\mathcal{O}\left(n\binom{n+2t+1}{1+t}^{\omega}\right)$$

field operations using Matrix-F5.

9.3 Applications

9.3.1 The McEliece cryptosystem

In [Mce78] McEliece proposed a cryptosystem based on the hardness of decoding a general linear code. In its original description, which remains unbroken until today, a random Goppa code disguised as a general linear code is used. Goppa codes know an efficient decoding algorithm. Knowledge of this decoding algorithm together with the permutation matrix used to disguise the code.
The system takes as input the parameters $n, t \in \mathbb{N}$ with t much smaller than n.
The private key consists of a random Goppa code of length n, dimension k and minimum distance at least $2t+1$. Hence, its error-correcting capability equals t. This code is described by a $k \times n$ generator matrix G. Associated with the code is an efficient decoding algorithm D_G. We generate a random $n \times n$ permutation matrix P and a random $k \times k$ non-singular matrix S which are both kept secret.
The public key is formed by permuting the coordinates of the code. As a second step, the rows of the generator matrix of this permuted code are scrambled to ensure that it is not in standard form, as this would reveal most of the bits of the plaintext. We then compute $\hat{G} = SGP$, which generates a code with the same minimum distance. Finally, we publish $\hat{G}$.
After setting up the key-pair, the algorithms below are used by anyone wishing to communicate in a secure way.

input : A plaintext message $m \in \mathbb{F}_q^k$, a $k \times n$ generator matrix $\hat{G}$, and a parameter t
output: A ciphertext $c \in \mathbb{F}_q^n$
begin
  Choose a vector $e \in \mathbb{F}_q^n$ of weight t randomly ;
  // The message is hidden by distorting it with e ;
  return $m\hat{G} + e$
end

Algorithm 18: McEliece encryption

There are two ways of attacking this system:

- Try to recover G, given $\hat{G}$, i.e., finding the secret code (structural attack).

input : A ciphertext $c \in \mathbb{F}_q^n$, a $\times$ permutation matrix P, a $\times$ non-singular matrix S, and an efficient decoding algorithm D_G
output: A plaintext message $m \in \mathbb{F}_q^k$
begin
  Calculate $cP^{-1} = mSG + eP^{-1}$;
  Compute $mS = D_G(cP^{-1})$;
  $m = (mS)S^{-1}$;
  return m
end

Algorithm 19: McEliece decryption

- Try to recover m given c without learning anything about the secret code (general attack).

Now, we have seen how we can set up a system of equations corresponding to the code described by $\hat{G}$ and solve this by means of a Gröbner basis computation to find m.

9.3.2 Finding the minimum distance

It is possible to find the minimum distance of the code by solving the system $J(t, r)$ with the parameter r set to 0. The idea is that the smallest t for which the system $J(t, 0)$ has a solution with nonzero u-component corresponds exactly to the minimum distance.

Theorem 9.3.1. *Let B be an MDS matrix with structure constant μ_l^{ij}. Let H be a parity check matrix for the code C such that $H = AB$ for some A. Let t be the smallest integer such that $J(t, 0)$ has a solution (u, v) with $u \neq 0$. Then $d(C) = t$.*

Proof. Let $0 \neq c \in C$ be a code word with $\mathrm{wt}(c) = d(C)$. Now, $c + (-c) = 0$ and $\mathrm{wt}(-c) = d(C)$ as well. Hence, lemma 9.1.17 says that there exists a v such that $(u(-c), v)$ is a zero of $J(d(C), 0)$. Since $-c \neq 0$ we also have that $u(-c) \neq 0$. Are there any solutions to $J(t, 0)$ with $t < d(C)$ and $u \neq 0$? Suppose there is one, say (u', v'). Then by lemma 9.1.18 there exists a unique e with $\mathrm{wt}(e) \leq t$ such that $u' = u'(e)$. Now, $u' \neq 0$ so $e \neq 0$. Moreover, $0 = r = c + e$ for some $c \in \mathbb{F}_{q^m} C$ for some $m \in \mathbb{Z}_{>0}$, hence $e = -c \in \mathbb{F}_{q^m} C$. But since $d(C) = d(\mathbb{F}_{q^m} C)$ we have found a code word of weight strictly smaller than the minimum weight, which is of course a contradiction. □

Example 9.3.2.
```
> C := GolayCode(GF(3), false);
> C;
[11, 6, 5] "Unextended Golay Code" Linear Code over GF(3)
Generator matrix:
[1 0 0 0 0 0 2 0 1 2 1]
[0 1 0 0 0 0 1 2 2 2 1]
[0 0 1 0 0 0 1 1 1 0 1]
[0 0 0 1 0 0 1 1 0 2 2]
[0 0 0 0 1 0 2 1 2 2 0]
[0 0 0 0 0 1 0 2 1 2 2]
>
> MyMinimumDistance(C);
5
```

Chapter 10

Linear codes as binomial ideals

It turns out that we can associate with a systematic linear code C a binomial ideal that is the sum of a toric ideal and a non-prime ideal in the case that the finite field is a prime field. We can easily read off the reduced Gröbner basis, G say, for this ideal with respect to any ordering from a generator matrix of C. Given G, encoding and decoding is reduced to multivariate division by G. This chapter closely follows the exposition by Schmidt [Sch14], but the idea goes back to [BQBTFMM08] and can be found in full detail in [MC13].

10.1 Toric ideals

We need some terminology that was not covered in the first chapter. Let $k[X] = k[x_1, \ldots, x_n]$.

Definition 10.1.1 (Binomial)**.** *A binomial in $k[X]$ is a polynomial that is the sum of two terms, i.e., it has the form $aX^u + bX^v$ where $a, b \in k$ and $u, v \in \mathbb{N}^n$. A binomial is said to be unitary if $a = 1$ and $b = -1$. A unitary binomial is said to be pure if $\gcd(X^u, X^v) = 1$.*

Definition 10.1.2 (Binomial ideal)**.** *An ideal in $k[X]$ is a binomial ideal if it is generated by binomials.*

A Gröbner basis for a binomial ideal relative to any monomial order consists entirely of binomials. This can be seen by taking the generating set, consisting of binomials, and applying Buchberger's algorithm to it.
There exists a subclass of binomial ideals, the toric ideals, that frequently make an appearance in various applied mathematics problems.

Definition 10.1.3 (Toric ideal)**.** *A binomial ideal that is prime is called a toric ideal.*

Toric ideals often arise as follows. Let $A \in \mathbb{Z}^{d \times n}$ be an integer matrix. Each column vector $a_i = (a_{i1}, \ldots, a_{id})^T$ of A can be identified with a so-called Laurent monomial in

$$k[y_1, \ldots, y_d, y_1^{-1}, \ldots, y_d^{-1}] =: k[Y, Y^{-1}],$$

the ring of Laurent polynomials, as follows:

$$a_i \mapsto Y^{a_i} = y_1^{a_{i1}} \cdot \cdots \cdot y_d^{id} \text{ for } 1 \leq i \leq n$$

Let $\phi : k[X] \to k[Y, Y^{-1}]$ be the k-algebra homomorphism given by

$$\phi(x_i) = Y^{a_i} \text{ for } 1 \leq i \leq n$$

We will associate the kernel of ϕ with the matrix A. It turns out that this is a toric ideal.

Definition 10.1.4 (Toric ideal associated with A)**.** *The toric ideal associated with A, denoted I_A, is the kernel of ϕ.*

Now, let $u \in \mathbb{Z}^n$ and write $u^+ = (\max\{0, u_1\}, \ldots, \max\{0, u_n\})$ and $u^- = (-u)^+$. Then we can write $u = u^+ - u^-$ in a unique way.

Proposition 10.1.5. *I_A is spannend as a k-vector space by*

$$I_A = \langle X^{u^+} - X^{u^-} : u \in \mathbb{Z}^n, Au = 0 \rangle$$

Moreover, I_A is prime. It follows that I_A is toric.

Proof. We have that $k[X]/I_A \simeq k[Y^{a_1}, \ldots, Y^{a_n}]$ and the right hand side is an integral domain. It follows that I_A is a prime ideal. Next, let $u \in \mathbb{Z}^n$ with $Au = 0$. The latter implies that $Au^+ = Au^-$. In turn, this implies that

$$\phi(X^{u^+} - X^{u^-}) = \phi(X^{u^+}) - \phi(X^{u^-}) = Y^{Au^+} - Y^{Au^-} = Y^{Au^-} - Y^{Au^-} = 0$$

and so $X^{u^+} - X^{u^-} \in I_A$. For the other direction, let $f \in I_A$ and suppose that f can not be written as a k-linear combination of the $X^{u^+} - X^{u^-}$ and has minimal leading monomial among all polynomials with this property. By assumption $\phi(f) = 0$. In particular, $\phi(\mathrm{lm}(f))$ must cancel, hence there exists some other monomial x^β appearing in f with $x^\beta < \mathrm{lm}(f)$ such that $f' := \phi(\mathrm{lm}(f)) = \phi(x^\beta)$. The polynomial $f - \mathrm{lm}(f) + x^\beta$ can also not be written as a k-linear combination of the $X^{u^+} - X^{u^-}$. But $\mathrm{lm}(f') < \mathrm{lm}(f)$, which is a contradiction. □

This shows a connection between the toric ideal and the kernel of the matrix A.
From now on we assume that the entries of A are non-negative integers. This simplifies things somewhat. In particular, $I_A = I$ is equal to the elimination ideal $\langle x_i - Y^{a_i} : 1 \leq i \leq n \rangle \cap k[X]$. Hence, we can find a Gröbner basis for I by considering the ideal $J = \langle x_i - Y^{a_i} : 1 \leq i \leq n \rangle$ in $k[X, Y]$ and computing a Gröbner basis G for J relative to the lexicographic order with $x_1 > \cdots > x_n > y_1 > \cdots > y_d$. The elimination theorem then tells us that $G \cap k[X]$ is a Gröbner basis for $J \cap k[X] = I$.
Let p be a prime number. We can associate with the toric ideal I_A a binomial ideal

$$I_{A,p} = I_A + \langle x_i^p - 1 : 1 \leq i \leq n \rangle$$

10.2 The code ideal

Let C be a systematic $[n, k]$ code over $\mathbb{F}_q$. We can view $\mathbb{F}_q = \mathbb{F}_{p^l}$ as a finite dimensional vector space over its prime field $\mathbb{F}_p$. Let $\{b_1, \ldots, b_l\}$ be any basis for $\mathbb{F}_q$ as an $\mathbb{F}_p$ vector space. Then there exists a unique vector space isomorphism

$$\psi : \mathbb{F}_q \to \mathbb{F}_p^l$$

such that $\psi(b_i) = e_i$ for $1 \leq i \leq l$. It follows that we can identify elements of $\mathbb{F}_q$ with vectors with entries in $\mathbb{F}_p$. Now, we can represent the elements of $\mathbb{F}_p$ by the integers in the set $\{0, \ldots, p-1\}$ using the ring isomorphism $\mathbb{F}_p \to \mathbb{Z}/p\mathbb{Z}$. As a result, vectors with entries in $\mathbb{F}_p$ are mapped to integral vectors. Formally, we have a map

$$f : \mathbb{F}_p^l \to \mathbb{Z}^l$$

The composition of f and ψ yields a mapping converting elements from $\mathbb{F}_q$ to integral vectors. Applying this map componentwise to vectors of elements in $\mathbb{F}_q$ yields a mapping

$$g : \mathbb{F}_q^n \to \mathbb{Z}^{ln}$$

Let X denote the vector of the n variables $X_1, \ldots, X_n$, and, for $1 \leq i \leq n$ let X_i be decomposable as $x_{i1}, \ldots, x_{il}$. Let $k[X]$ be the polynomial ring in nl variables. We can identify vectors in $\mathbb{F}_q^n$ with monomials in $k[X]$ as follows. Let $u \in \mathbb{F}_q^n$, then the corresponding monomial is $x^{g(u)}$.

$$\mathbb{Z}^{ln} \to \mathbb{F}_p^{ln}$$

We can then define an ideal relative to C:

Definition 10.2.1 (Ideal associated with C)**.** *We will associate with C an ideal that is a sum of two binomial ideals*

$$I_C = \langle X^{g(c)} - X^{g(c')} : c - c' \in C \rangle \cup \langle x_i^p - 1 : 1 \leq i \leq n \rangle$$

The condition $c - c' \in C$ is equivalent to stating that $Hc = Hc' \pmod q$. Hence, theorem shows that $I_C = I_{A,p}$ for some matrix A over $\mathbb{Z}$ such that $H = A \otimes_{\mathbb{Z}} \mathrm{Id}_{\mathbb{F}_p}$ where H is a parity check matrix for C over $\mathbb{F}_p$. This shows that we can write I_C as the sum of a toric ideal and a non-prime ideal.

Proposition 10.2.2. *Let C be an $[n, k]$ code over $\mathbb{F}_q$ and $G = (g_{ij})$ its generator matrix, and let $\{b_1, \ldots, b_l\}$ be a basis for $\mathbb{F}_q$ as an $\mathbb{F}_p$-vector space. The ideal I_C is generated by*

$$I_C = \langle X^{b_i g_j} - 1 : 1 \leq i \leq l, 1 \leq j \leq k \rangle \cup \langle x_{ij}^p - 1 : 1 \leq i \leq n, 1 \leq j \leq l \rangle$$

Computing a Gröbner basis, as we have seen, normally is a very costly operation. However, the ideal thus defined has a reduced Gröbner basis that can easily be constructed by looking at the entries of the generator matrix over the prime field. To this end, let G be the $k \times n$ generator matrix for C

$$G = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{pmatrix}$$

Since C is assumed to be systematic, G is in row reduced echelon form. Recall that we have a basis $B = \{b_1, \ldots, b_l\}$ for $\mathbb{F}_q$ as an $\mathbb{F}_p$ vector space. We construct a new matrix $G_{ext} \in \mathbb{F}_p^{kl \times nl}$ derived from G in the following way:

$$G_{ext} = \begin{pmatrix} \psi(b_1 g_{11}) & \psi(b_1 g_{12}) & \ldots & \psi(b_1 g_{1n}) \\ \vdots & \vdots & & \vdots \\ \psi(b_r g_{11}) & \psi(b_r g_{12}) & \ldots & \psi(b_r g_{1n}) \\ \psi(b_1 g_{21}) & \psi(b_1 g_{22}) & \ldots & \psi(b_1 g_{2n}) \\ \vdots & \vdots & & \vdots \\ \psi(b_r g_{k1}) & \psi(b_r g_{k2}) & \ldots & \psi(b_r g_{kn}) \end{pmatrix}$$

Clearly, G_{ext} is in row reduced echelon form as well, i.e., $G_{ext} = (e_{ij} - m_{ij})$. In the following, we will index the rows of G_{ext} with the set $\{11, \ldots, 1r, 21, \ldots 2r, \ldots, k1, \ldots, kr\}$ in the obvious way.

Theorem 10.2.3. *Let $>$ be the lexicographic order with $x_1 > \cdots > x_n$. Then the reduced Gröbner basis (relative to $>$) for I_C has the shape*

$$G = \{x_{ij} - X^{m_{ij}} : 1 \leq i \leq k, 1 \leq j \leq l\} \cup \{x_{ij}^p - 1 : k+1 \leq i \leq n, 1 \leq j \leq l\}$$

Proof. Observe that the leading terms of the generators are relatively prime. By Buchberger's first criterion $\langle G \rangle$ is a Gröbner basis. Hence, we need only to show that $\langle G \rangle = I_C$. By inspection, we have that $\langle G \rangle \subseteq I_C$. Conversely, we need to show that $I_C \subseteq \langle G \rangle$. It suffices to show that any binomial in the generating set of I_C can be written in terms of G. Consider the binomial $X^{b_i g_j} - 1$ for some $1 \leq i \leq l, 1 \leq j \leq k$. Write $m = g_i - e_i$ and $J = \langle x_{ij}^p - 1 : k+1 \leq i \leq n, 1 \leq j \leq l \rangle$. Since G is in standard form, the first k positions of m are zero. Now, the claim is that

$$X^{b_j m}(x_{ij} - X^{m_{ij}}) = x^{b_j g_i} - 1 \pmod{J}$$

Indeed,

$$X^{b_j m} x_{ij} = X^{b_j m} X_i^{b_j} = X^{b_j m - b_j e_i} = X^{b_j g_i}$$

. Moreover, $X^{\psi(b_j m)} X^{m_{ij}} = X^{\psi(b_j g_i) + \psi(b_j e_i) + e_{ij} - \psi(b_j g_i)} = X^{e_{ij} - \psi(b_j e_i)} = 1 \mod J$. The claim follows. Next, consider the binomial $x_{ij}^p - 1$ for some $1 \leq i \leq n, 1 \leq j \leq l$. The claim is that

$$x_{ij}^p - 1 = \left(\sum_{i=0}^{p-1} x_{ij}^{p-1-i} X^{i m_{ij}} \right) (x_{ij} - X^{m_{ij}}) \mod J$$

All terms cancel out, except for x_{ij}^p and $X^{p m_{ij}}$, but $X^{p m_{ij}} = 1 \mod J$ and thus the claim follows. This proves the statement. □

Example 10.2.4. *Consider the code ternary C with parameters $[7, 2, 5]$ and generator matrix*

$$\begin{pmatrix} 1 & 0 & 1 & 2 & 1 & 1 & 1 \\ 0 & 1 & 2 & 2 & 1 & 0 & 2 \end{pmatrix}$$

```
> P<[X]> := PolynomialRing(GF(3), 7);
> I := [X[1]-X[3]^2*X[4]*X[5]^2*X[6]^2*X[7]^2, X[2]-X[3]*X[4]*X[5]^2*X[7]] cat\
 [X[i]^3-1 : i in [3..7]];
> I;
[
    2*X[3]^2*X[4]*X[5]^2*X[6]^2*X[7]^2 + X[1],
    2*X[3]*X[4]*X[5]^2*X[7] + X[2],
    X[3]^3 + 2,
    X[4]^3 + 2,
    X[5]^3 + 2,
    X[6]^3 + 2,
    X[7]^3 + 2
]
> GroebnerBasis(I);
[
    X[1] + 2*X[3]^2*X[4]*X[5]^2*X[6]^2*X[7]^2,
    X[2] + 2*X[3]*X[4]*X[5]^2*X[7],
    X[3]^3 + 2,
```

```
    X[4]^3 + 2,
    X[5]^3 + 2,
    X[6]^3 + 2,
    X[7]^3 + 2
]
```

Corollary 10.2.5. *Let $J = \{j_1, \ldots, j_k\}$ be an information set and let the generator matrix G for C be in row reduced echelon form with respect to the columns indexed by J, and let $>$ be any term order with $x_1 > \cdots > x_n$. Then the reduced Gröbner basis (relative to $>$) for I_C has the shape*

$$G = \{x_{ij} - X^{m_{ij}} : i \in J, 1 \leq j \leq r\} \cup \{x_{ij}^p - 1 : i \in \{1, \ldots, n\} \setminus J, 1 \leq j \leq r\}$$

10.3 A heuristic for decoding general linear codes

Now, suppose that C has error-correcting capability τ.

Proposition 10.3.1. *The standard monomials yield a transversal of the quotient space $\mathbb{F}_q^n/C$.*

Proof. Every standard monomial corresponds to a unique coset in a one-to-one way. Indeed, suppose there existed standard monomials X^u and $X^{u'}$ such that $u + C = u' + C$. Then $u - u' \in C$ and therefore $X^u - X^{u'} \in I_C = \langle G \rangle$. Without loss of generality, assume that $\mathrm{lt}(X^u - X^v) = X^u$. It follows that $X^u \in \mathrm{lt}(I)$, which contradicts the assumption X^u is standard. □

Proposition 10.3.2. *Let C be an $[n, k, d]$ code over $\mathbb{F}_q$ with error-correcting capability $\tau = \lfloor \frac{d-1}{2} \rfloor$, and let G be the reduced Gröbner basis relative to a degree compatible order for the ideal I_C. Let $r \in \mathbb{F}_q^n$ be a received word. If the normal form of $X^{g(r)}$ with respect to G is a monomial $X^{g(e)}$ such that $\mathrm{wt}(e) \leq \tau$, then $r - e$ is the unique closest codeword to r.*

Proof. Let $r = c + e$ for a codeword c and error e with $\mathrm{wt}(e) \leq \tau$. Let $X^{g(r)} \operatorname{rem} G = X^{g(f)}$ for some f. Hence, $X^{g(r)} - X^{g(f)} \in I_C$. It follows that $r - f \in C$. By assumption, $\mathrm{wt}(f) \leq \tau$, so $d(r, r - f) = \mathrm{wt}(f) \leq \tau$. Since there exists a unique codeword in the ball of radius τ around r it follows that $r - f = c$. Hence $e = f$ and $r - e$ is the unique closest codeword to r. □

From the proposition, a decoding algorithm can be deduced: convert the received word to a monomial and compute the remainder with respect to the Gröbner basis. However, this remainder will often not correspond to the error vector. In fact, the remainders for which this hold are exactly the non-standard monomials.
Fortunately, there is a way to improve the algorithm which is based on the following observation: if we multiply r by any nonzero scalar a, then the minimum weight word in the coset $ar + C$ is equal to ae. Hence, finding e is equivalent to finding ae. However, X^{ae} might be a standard monomial.
The following proposition shows when 20 fails to decode.

Proposition 10.3.3. *Let $r = c + e$ be the received word such that the weight of the error vector is at most τ. Algorithm 20 will fail to find the correct error vector if and only if $X^{a(r-c)}$ is a non-standard monomial for every non-zero $a \in \mathbb{F}_q$.*

Proof. See [Sch14] proposition 5.3.7. □

input : A received word $r \in \mathbb{F}_q^n$, a reduced Gröbner basis G relative to $>$ for I_C, the error-correcting capability τ
output: Either c or *fail*
begin
  $A = [a_1, \ldots, a_{q-1}]$, the nonzero elements of $\mathbb{F}_q$;
  for $i := 1$ **to** $q - 1$ **do**
    $r := A[i] * r$;
    // create vector ;
    $X^e := NormalForm(X^r, G)$;
    if $\mathrm{wt}(e) \leq \tau$ **then**
      $c := r - e$;
      $c := A[i]^{(-1)} * c$;
      return c
    end
  end
  return *fail*
end

Algorithm 20: Heuristic for decoding a general code

Compared to syndrome decoding, where we need to store $\mathcal{O}(p^{n-k})$ elements, we only need to store the Gröbner basis, consisting of $\mathcal{O}(n)$ elements.

Example 10.3.4. *Consider the code ternary C with parameters $[7, 2, 5]$ and generator matrix*

$$\begin{pmatrix} 1 & 0 & 1 & 2 & 1 & 1 & 1 \\ 0 & 1 & 2 & 2 & 1 & 0 & 2 \end{pmatrix}$$

Now, suppose the codeword $c = (1, 2, 2, 0, 0, 1, 2)$ has been transmitted and the word $r = (0, 1, 2, 0, 0, 1, 2)$ has been received. So two errors have been introduced.

```
> P<[X]> := PolynomialRing(GF(3), 7, "grevlex");
> I := [X[1]-X[3]^2*X[4]*X[5]^2*X[6]^2*X[7]^2, X[2]-X[3]*X[4]*X[5]^2*X[7]] cat\
 [X[i]^3-1 : i in [3..7]];
> I;
[
    2*X[3]^2*X[4]*X[5]^2*X[6]^2*X[7]^2 + X[1],
    2*X[3]*X[4]*X[5]^2*X[7] + X[2],
    X[3]^3 + 2,
    X[4]^3 + 2,
    X[5]^3 + 2,
    X[6]^3 + 2,
    X[7]^3 + 2
]
> G := GroebnerBasis(I);
> r := VectorSpace(GF(3), 7) ! [0,1,2,0,0,1,2];
> t := 2;
> HeuristicDecode(P, G, t, r);
Trying with a =  1
```

```
Found the wrong error vector:  (0 0 0 1 2 1 0)
Trying with a =  2
Found the right error vector:  (1 1 0 0 0 0 0)
The codeword is:
(1 2 2 0 0 1 2)
>
```

Chapter 11

Experimental results

This chapter contains a number of experiments related to the decoding method based on quadratic equations. In particular, we do not compare this method to any of the other methods, as this has already been considered by Bulygin in [Bul09], but we want to figure out more about the running time of the decoding algorithm. The bottleneck in the algorithm is, of course, the computation of a Gröbner basis for the system. What is the maximum degree reached during a Gröbner basis computation? This number basically determines the complexity of most contemporary Gröbner basis algorithms. In particular, it is the size of the largest matrix appearing in any linear algebra-based Gröbner basis algorithm.

Magma has functions for returning codes with the best known parameters. In particular, fixing any two among the parameters length, dimension, and minimum distance it returns the code with the best known value for the third. Here best means shortest length, or largest minimum distance, or largest dimension.

The following table considers binary codes. It shows the time it takes to compute a Gröbner basis for the system $J(1, r)$ when we keep the information rate $R = \frac{k}{n}$ fixed at the constant $\frac{1}{2}$ while increasing both k and n and asking Magma for the code with the largest minimum distance among all $[n, k]$-codes.

Parameters	Time (s)
$[8, 4, 4]$	0.000
$[16, 8, 5]$	0.000
$[32, 16, 8]$	0.010
$[64, 32, 12]$	0.010
$[128, 64, 22]$	0.240
$[256, 128, 38]$	3.570

The following table considers binary codes. It shows the time it takes to compute a Gröbner basis for the system $J(1, r)$ when we keep the relative distance $\delta = \frac{d}{n}$ fixed at the constant $\frac{1}{2}$ (it cannot be more) while increasing both d and n and asking Magma for the code with the largest dimension among all $[n, -, d]$-codes.

Parameters	Time (s)
$[8, 4, 4]$	0.000
$[16, 5, 8]$	0.000
$[32, 6, 16]$	0.000
$[64, 7, 32]$	0.010
$[128, 8, 64]$	0.220
$[256, 9, 128]$	3.380

The only thing that we are able to gather from the tables above is that the length of the code plays a big role in the running time of the algorithm. This is to be expected, as it determines the size of the largest matrix appearing in the computation of a Gröbner basis.

This experiment shows the maximum degree reached in a computation of the Gröbner basis for the homogenized version of $J(t, r)$. We see what happens for an increasing number of errors which occur at randomly selected positions in a randomly generated codeword. A bar denotes that we aborted the computation because it took too much time.

	d_{reg}			
Parameters	$t = 1$	$t = 2$	$t = 3$	$t = 4$
$[32, 4, 9]$	2	2	2	2
$[32, 4, 10]$	2	2	2	2
$[32, 4, 11]$	2	2	2	2
$[32, 4, 12]$	2	2	2	2
$[32, 4, 13]$	2	2	2	2
$[32, 8, 6]$	2	2	-	-
$[32, 8, 7]$	2	2	3	2
$[32, 8, 8]$	2	2	3	2
$[32, 8, 9]$	2	2	3	3
$[32, 8, 10]$	2	2	3	3
$[32, 16, 4]$	3	3	-	-
$[32, 16, 5]$	3	3	-	-
$[32, 24, 3]$	3	-	-	-
$[64, 32, 8]$	3	3	4	-
$[64, 32, 9]$	3	3	4	5
$[64, 48, 3]$	3	-	-	-
$[128, 16, 40]$	2	2	2	2
$[128, 16, 41]$	2	2	2	2
$[128, 64, 14]$	3	3	4	-
$[128, 64, 15]$	3	3	4	-
$[128, 64, 16]$	3	3	4	-
$[128, 96, 7]$	3	3	4	-

The first thing we notice is that the degree of regularity does not seem to depend on where the errors occur. The second thing we notice is that it does seem to depend on the rate R. Clearly when R increases the degree of regularity increases as well.

Recall that $J(t, r)$ is a system of $n + (n - k) = n + r$ polynomials, r of which are linear and n of which are quadratic, in $n + t$ variables. The ratio between linear polynomials and quadratic polynomials is given by $\frac{n-k}{n} = 1 - R$. As the rate of the code increases, this ratio becomes smaller and the performance becomes worse. One could conjecture that increasing the number of linear polynomials in the system positively influences the complexity. This might be explained by the fact that Magma's Gröbner basis algorithm proceeds by degree.

Bibliography

[ABF09] Daniel Augot, Magali Bardet, and Jean-Charles Faugère. On the decoding of binary cyclic codes with the Newton's identities. *Journal of Symbolic Computation*, 44(12):1608–1625, December 2009.

[Alb10] Martin Albrecht. *Algorithmic Algebraic Techniques and their Application to Block Cipher Cryptanalysis*. PhD thesis, Royal Holloway, University of London, 2010.

[AM69] Michael Atiyah and Ian Grant Macdonald. Introduction to commutative algebra, 1969.

[Bar04] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie*. Theses, Université Pierre et Marie Curie - Paris VI, December 2004.

[Ber84] E. R. Berlekamp. *Algebraic coding theory*. Aegean Park Press, Laguna Hills, CA, USA, 1984.

[BFS15] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of the F5 Gröbner basis algorithm. *J. Symb. Comput.*, 70(C):49–70, September 2015.

[BFSY05] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Bo-Yin Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *In MEGA05, 2005. Eighth International Symposium on Effective Methods in Algebraic Geometry*, pages 1–14, 2005.

[BMvT06] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inf. Theor.*, 24(3):384–386, September 2006.

[BP09] Stanislav Bulygin and Ruud Pellikaan. Bounded distance decoding of linear error-correcting codes with Gröbner bases. *J. Symb. Comput.*, 44(12):1626–1643, December 2009.

[BQBTFMM08] M. Borges-Quintana, M. A. Borges-Trenard, P. Fitzpatrick, and E. Martínez-Moro. Gröbner bases and combinatorics for binary codes. *Applicable Algebra in Engineering, Communication and Computing*, 19(5):393–411, 2008.

[BRC60] R.C. Bose and D.K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68 – 79, 1960.

[Bul09] Stanislav Bulygin. *Polynomial system solving for decoding linear codes and algebraic cryptanalysis*. PhD thesis, University of Kaiserslautern, 2009.

[BWK93] Thomas Becker, Volker Weispfenning, and Heinz Kredel. *Gröbner Bases: A Computational Approach to Commutative Algebra*. Springer-Verlag, London, UK, UK, 1993.

[CCS99] Arjeh M. Cohen, Hans Cuypers, and Hans Sterk. *Some tapas of computer algebra*. Algorithms and computation in mathematics. Springer Verlag, New York, Berlin, Heidelberg, 1999.

[CLO15] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer Publishing Company, Incorporated, 4th edition, 2015.

[DS13] Jintai Ding and Dieter Schmidt. *Solving Degree and Degree of Regularity for Polynomial Systems over a Finite Fields*, pages 34–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[Dub90] Thomas W. Dubé. The structure of polynomial ideals and gröbner bases. *SIAM Journal on Computing*, 19(4):750–775, August 1990.

[Ede13] Christian Eder. An analysis of inhomogeneous signature-based Gröbner basis computations. *J. Symb. Comput.*, 59:21–35, December 2013.

[EF15] Christian Eder and Jean-Charles Faugère. A survey on signature-based Gröbner basis computations. *ACM Commun. Comput. Algebra*, 49(2):61–61, August 2015.

[Eis95] David Eisenbud. *Commutative algebra: with a view toward algebraic geometry*. Graduate texts in mathematics. Springer, New York, Berlin, Heildelberg, 1995. Rimpr. corr. en 1996. Autres tirages : 1999, 2004.

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Grb̈ner bases (F4). *Journal of Pure and Applied Algebra*, 139(13):61 – 88, 1999.

[Fau02] Jean Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, pages 75–83, New York, NY, USA, 2002. ACM.

[FMLG89] Jean-Charles Faugère, Teo Mora, Daniel Lazard, and P. Gianni. Efficient computation of zero-dimensional Gröbner bases by change of ordering. Technical Report 89-52, Universit Denis Diderot (Paris), 1989.

[FSEDS11] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1,1): Algorithms and complexity. *J. Symb. Comput.*, 46(4):406–437, April 2011.

[Ful] William Fulton. Algebraic curves: An introduction to algebraic geometry.

[Gal12] François Le Gall. Faster algorithms for rectangular matrix multiplication. *CoRR*, abs/1204.1111, 2012.

[Giu84] M. Giusti. Some effectivity problems in polynomial ideal theory. *EUROSAM 84*, 174:159–171, 1984.

[GMN+91] Alessandro Giovini, Teo Mora, Gianfranco Niesi, Lorenzo Robbiano, and Carlo Traverso. "One sugar cube, please"; or selection strategies in the buchberger algorithm. In *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*, ISSAC '91, pages 49–54, New York, NY, USA, 1991. ACM.

[GVIW16] Shuhong Gao, Frank Volny IV, and Mingsheng Wang. A new framework for computing Gröbner bases. *Mathematics of computation*, 85(297):449–465, January 2016.

[GZ61] D.C. Gorenstein and N. Zierler. A class of error-correcting codes in p^m symbols. *SIAM*, 9:207 – 214, 1961.

[Hoc59] A. Hocquenghem. Codes Correcteurs d'Erreurs. *Chiffres (Paris)*, 2:147–156, September 1959.

[Huo13] Louise Huot. *Polynomial systems solving and elliptic curve cryptography.* Theses, Université Pierre et Marie Curie - Paris VI, December 2013.

[KR05] Martin Kreuzer and Lorenzo Robbiano. *Computational commutative algebra. 2.* Springer-Verlag, Berlin, 2005.

[KR08] Martin Kreuzer and Lorenzo Robbiano. *Computational Commutative Algebra 1.* Springer Publishing Company, Incorporated, 2008.

[Laz83] Daniel Lazard. Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations. In *Proceedings of the European Computer Algebra Conference on Computer Algebra*, EUROCAL '83, pages 146–156, London, UK, UK, 1983. Springer-Verlag.

[Mac94] F.S. Macaulay. *The Algebraic theory of modular systems.* Cambridge mathematical library. Cambridge University Press, Cambridge, New York, Melbourne, 1994.

[MC13] Irene Márquez-Corbella. *Combinatorial Commutative Algebra Approach to Complete Decoding.* PhD thesis, Institute of Mathematics, University of Valladolid, 2013.

[Mce78] Robert J. Mceliece. A public-key cryptosystem based on algebraic coding theory. Technical report, Jet Propulsion Lab Deep Space Network Progress report, 1978.

[MS77] Florence Jessie MacWilliams and N. J. A. Neil James Alexander Sloane. *The theory of error correcting codes.* North-Holland mathematical library. North-Holland Pub. Co. New York, Amsterdam, New York, 1977. Includes index.

[OS05] Emmanuela Orsini and Massimiliano Sala. Correcting errors and erasures via the syndrome variety. *Journal of Pure and Applied Algebra*, 200(12):191 – 226, 2005.

[Pet60] W.W. Peterson. Encoding and error-correction procedures for the Bose-Chaudhuri codes. *IRE Trans. Inform. Theory*, IT-6:459 – 470, 1960.

[RS12] Bjarke Hammersholt Roune and Michael Stillman. Practical Gröbner basis computation. In *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*, ISSAC '12, pages 203–210, New York, NY, USA, 2012. ACM.

[Sch14] Natalia Schmidt. *Gröbner Bases in Coding Theory*. PhD thesis, Hamburg University of Technology, 2014.

[SHWL16] Yao Sun, Zhenyu Huang, Dingkang Wang, and Dongdai Lin. An improvement over the GVW algorithm for inhomogeneous polynomial systems. *Finite Fields and Their Applications*, 41:174–192, June 2016.

[SKHN75] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. A method for solving the key equation for decoding goppa codes. *Information and Control*, 27:287–99, 1975.

[Spa12] Pierre-Jean Spaenlehauer. *Solving multi-homogeneous and determinantal systems: algorithms, complexity, applications*. Theses, Université Pierre et Marie Curie (Univ. Paris 6), October 2012.

[Sto00] Arne Storjohann. *Algorithms for matrix canonical forms*. PhD thesis, ETH Zürich, 2000.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13(4):354–356, August 1969.

[Sud] Madhu Sudan. 6.S897 Algebra and Computation the complexity of the ideal membership problem. http://people.csail.mit.edu/madhu/ST15/scribe/lect21.pdf. Accessed: 2017-01-24.

[Sva14] Jules Svartz. *Solving zero-dimensional structured polynomial systems*. Theses, Université Pierre et Marie Curie - Paris VI, October 2014.

[Vac15] Tristan Vaccon. Matrix-F5 algorithms and tropical Gröbner bases computation. In *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '15, pages 355–362, New York, NY, USA, 2015. ACM.

[Var97] Alexander Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 92–109, New York, NY, USA, 1997. ACM.

[Ver16] Thibaut Verron. *Regularisation of Gröbner basis computations for weighted and determinantal systems, and an application to medical imagery.* Theses, Université Pierre et Marie Curie, September 2016.

[Vol11] Frank Volny. *New algorithms for computing Groebner bases.* PhD thesis, Clemson University, 2011.

Appendix A

Implementations in Magma

A.1 The Matrix-F5 algorithm

Listing A.1: The Macaulay matrix data type

```
declare type MtrxMcly;
declare attributes MtrxMcly: Signatures;
declare attributes MtrxMcly: Polynomials;
declare attributes MtrxMcly: Ring;

intrinsic MacaulayMatrix(P::RngMPol, d::RngIntElt : F := []) -> MacaulayMatrix
{ Constructor }
    M := New(MtrxMcly);

    M`Signatures := [];
    for k := 1 to #F do
        Append(~M`Signatures, <k, P ! 1, k>);
    end for;
    M`Polynomials := F;

    M`Ring := P;

    return M;
end intrinsic;

intrinsic CoefficientMatrix(M::MtrxMcly) -> Mtrx, ModTupRngElt
{ return the coefficient matrix associated with the polynomials }
    P := M`Ring;
    coefficients := [];
    monomials := [];

    for f in M`Polynomials do
        c, m := CoefficientsAndMonomials(f);
        Append(~coefficients, c);
        Append(~monomials, m);
    end for;

    sortedMonomials := Reverse(Sort(Setseq(Seqset(&cat monomials))));

    dict := AssociativeArray(sortedMonomials);
    col := 1;
    for mon in sortedMonomials do
```

```
38          dict[mon] := col;
39          col +:= 1;
40      end for;
41
42      A := ZeroMatrix(P, #M`Polynomials, #sortedMonomials);
43      for i := 1 to #M`Polynomials do
44          for j := 1 to #coefficients[i] do
45              A[i, dict[monomials[i,j]]] := coefficients[i, j];
46          end for;
47      end for;
48
49      return A, Vector(P, sortedMonomials);
50  end intrinsic;
51
52  intrinsic Print(M::MtrxMcly)
53  {Print X}
54      print CoefficientMatrix(M);
55  end intrinsic;
56
57  intrinsic NumberOfRows(M::MtrxMcly) -> RngIntElt
58  { returns the number of rows of M }
59      return #M`Polynomials;
60  end intrinsic;
61
62  intrinsic AddPolynomial(~M::MtrxMcly, f::RngMPolElt, s::Tup)
63  { Add the row representation of f to M }
64      Append(~M`Signatures, s);
65      Append(~M`Polynomials, f);
66  end intrinsic;
67
68  intrinsic ExtractPolynomial(M::MtrxMcly, i::RngIntElt) -> RngMPolElt
69  { Returns the polynomial represented by the i'th row of M }
70      return M`Polynomials[i];
71  end intrinsic;
72
73  intrinsic ExtractSignature(M::MtrxMcly, i::RngIntElt) -> Tup
74  { Returns the signature of row i }
75      return M`Signatures[i];
76  end intrinsic;
77
78  intrinsic Copy(M::MtrxMcly) -> MtrxMcly
79      { Returns a copy of this }
80
81      N := New(MtrxMcly);
82      N`Polynomials := M`Polynomials;
83      N`Signatures := M`Signatures;
84      N`Ring := M`Ring;
85
86      return N;
87  end intrinsic;
88
89  intrinsic Reduce(~M::MtrxMcly)
90      { Compute the row echelon form of M without permuting any rows }
91      A, v := CoefficientMatrix(M);
92
93      m := NumberOfRows(A);
94      n := NumberOfColumns(A);
```

```
95
96      for j := 1 to n do
97          for i := 1 to m do
98              if A[i, j] ne 0 then
99                  b := false;
100                 for k := 1 to j-1 do
101                     if A[i, k] ne 0 then
102                         b := true;
103                     end if;
104                 end for;
105
106                 if b then
107                     continue;
108                 end if;
109
110                 A := MultiplyRow(A, 1/A[i,j], i);
111                 for k := i+1 to m do
112                     if A[k, j] ne 0 then
113                         A := AddRow(A, -A[k, j], i, k);
114                     end if;
115                 end for;
116
117                 break;
118             end if;
119         end for;
120     end for;
121
122     M`Polynomials := ElementToSequence(v*Transpose(A));
123 end intrinsic;
```

Listing A.2: The Matrix-F5 algorithm

```
1  MatrixF5 := function(F, D)
2      R := Parent(F[1]);
3      vars := MonomialsOfDegree(R, 1);
4
5      G := F;
6      H := [];
7      M := AssociativeArray();
8      for d := 1 to D do
9          M[d] := MacaulayMatrix(R, d);
10         for i := 1 to #F do
11             if Degree(F[i]) eq d then
12                 AddPolynomial(~M[d], F[i], <i, R ! 1, NumberOfRows(M[d])+1>);
13             elif Degree(F[i]) lt d then
14                 for s in [s : s in M[d-1]`Signatures | s[1] eq i] do
15                     index := s[1];
16                     mon := s[2];
17                     row := s[3];
18
19                     max := 1;
20                     for var in vars do
21                         if Degree(mon, var) ne 0 then
22                             max := var;
23                             break;
24                         end if;
25                     end for;
26
```

```
                    for var in Reverse(Setseq(vars)) do
                        if var lt max then
                            continue;
                        end if;

                        found := false;

                        for h in H do
                            if h[1] eq i and IsDivisibleBy(var*mon, h[2]) then
                                found := true;
                                break;
                            end if;
                        end for;

                        if not found then
                            for t in [t : t in M[d-Degree(F[i])]`Signatures | t[1] lt i] do
                                index2 := t[1];
                                mon2 := t[2];
                                row2 := t[3];

                                if LeadingMonomial(ExtractPolynomial(M[d-Degree(F[i])], row2)) eq var*mon then
                                    found := true;
                                    break;
                                end if;
                            end for;
                        end if;

                        if not found then
                            AddPolynomial(~M[d], var*ExtractPolynomial(M[d-1], row), <index, var*mon, NumberOfRows(M[d])+1>);
                        end if;
                    end for;
                end for;
            end if;
        end for;

        T := Copy(M[d]);
        Reduce(~M[d]);

        pols := [];
        sigs := [];
        k := 1;

        for row := 1 to NumberOfRows(M[d]) do
            f := ExtractPolynomial(M[d], row);
            s := ExtractSignature(M[d], row);
            if f eq 0 then // found a syzygy
                Append(~H, <s[1], s[2]>);
                print "Row with signature ", s, "reduced to zero.";
            else
                Append(~pols, f);
                Append(~sigs, <s[1], s[2], k>);
                k +:= 1;

                if LeadingMonomial(ExtractPolynomial(T, row)) ne
```

```
    LeadingMonomial(ExtractPolynomial(M[d], row)) then
                    Append(~G, f);
                end if;
            end if;
        end for;

        M[d]`Polynomials := pols;
        M[d]`Signatures := sigs;

        delete T;
    end for;

    return Reduce(G);
end function;

Preprocess := function(F, n)
    R := Parent(F[1]);

    S := []; // syzygies found
    m := Rank(R)-n; // number of variables of second block

    if #F gt (n-1) + (m-1) then
        error "Runtime error 'Preprocess': too many polynomials on input.";
    end if;

    for i := 2 to #F do
        if i gt m then
            M := MacaulayMatrix(R, m : F := Minors(Submatrix(JacobianMatrix(F
    [1..i-1]), [1..i-1], [n+1..n+m]), m));
            Reduce(~M);

            for row := 1 to NumberOfRows(M) do
                h := ExtractPolynomial(M, row);
                if h ne 0 then
                    Include(~S, <i, LeadingMonomial(h)>);
                end if;
            end for;
        end if;

        if i gt n then
            M := MacaulayMatrix(R, n : F := Minors(Submatrix(JacobianMatrix(F
    [1..i-1]), [1..i-1], [1..n]), n));

            Reduce(~M);

            for row := 1 to NumberOfRows(M) do
                h := ExtractPolynomial(M, row);
                if h ne 0 then
                    Include(~S, <i, LeadingMonomial(h)>);
                end if;
            end for;
        end if;
    end for;

    return S;
end function;

```

```
// MatrixF5 for bilinear system
MatrixF5Bilinear := function(F, D, n)
    R := Parent(F[1]);
    vars := MonomialsOfDegree(R, 1);

    G := F;
    H := Preprocess(F, n);
    M := AssociativeArray();
    for d := 1 to D do
        M[d] := MacaulayMatrix(R, d);
        for i := 1 to #F do
            if Degree(F[i]) eq d then
                AddPolynomial(~M[d], F[i], <i, R ! 1, NumberOfRows(M[d])+1>);
            elif Degree(F[i]) lt d then
                for s in [s : s in M[d-1]`Signatures | s[1] eq i] do
                    index := s[1];
                    mon := s[2];
                    row := s[3];

                    max := 1;
                    for var in vars do
                        if Degree(mon, var) ne 0 then
                            max := var;
                            break;
                        end if;
                    end for;

                    for var in Reverse(Setseq(vars)) do
                        if var lt max then
                            continue;
                        end if;

                        found := false;

                        for h in H do
                            if h[1] eq i and IsDivisibleBy(var*mon, h[2]) then
                                found := true;
                                break;
                            end if;
                        end for;

                        if not found then
                            for t in [t : t in M[d-Degree(F[i])]`Signatures | t[1] lt i] do
                                index2 := t[1];
                                mon2 := t[2];
                                row2 := t[3];

                                if LeadingMonomial(ExtractPolynomial(M[d-Degree(F[i])], row2)) eq var*mon then
                                    found := true;
                                    break;
                                end if;
                            end for;
                        end if;

                        if not found then
```

```
                                AddPolynomial(~M[d], var*ExtractPolynomial(M[d-1],
    row), <index, var*mon, NumberOfRows(M[d])+1>);
                            end if;
                        end for;
                    end for;
                end if;
            end for;

            T := Copy(M[d]);
            Reduce(~M[d]);

            pols := [];
            sigs := [];
            k := 1;

            for row := 1 to NumberOfRows(M[d]) do
                f := ExtractPolynomial(M[d], row);
                s := ExtractSignature(M[d], row);
                if f eq 0 then // found a syzygy
                    Append(~H, <s[1], s[2]>);
                    print "Row with signature ", s, "reduced to zero.";
                else
                    Append(~pols, f);
                    Append(~sigs, <s[1], s[2], k>);
                    k +:= 1;

                    if LeadingMonomial(ExtractPolynomial(T, row)) ne
    LeadingMonomial(ExtractPolynomial(M[d], row)) then
                        Append(~G, f);
                    end if;
                end if;
            end for;

            M[d]`Polynomials := pols;
            M[d]`Signatures := sigs;

            delete T;
        end for;

        return Reduce(G);
end function;

// MatrixF5 for zero-dimensional systems
MatrixF5ZeroDimensional := function(F)
        R := Parent(F[1]);
        vars := MonomialsOfDegree(R, 1);

        G := F;
        H := [];
        M := AssociativeArray();

        d := 1;
        repeat
            M[d] := MacaulayMatrix(R, d);
            for i := 1 to #F do
                if Degree(F[i]) eq d then
                    AddPolynomial(~M[d], F[i], <i, R ! 1, NumberOfRows(M[d])+1>);
```

```
245             elif Degree(F[i]) lt d then
246                 for s in [s : s in M[d-1]`Signatures | s[1] eq i] do
247                     index := s[1];
248                     mon := s[2];
249                     row := s[3];
250 
251                     max := 1;
252                     for var in vars do
253                         if Degree(mon, var) ne 0 then
254                             max := var;
255                             break;
256                         end if;
257                     end for;
258 
259                     for var in Reverse(Setseq(vars)) do
260                         if var lt max then
261                             continue;
262                         end if;
263 
264                         found := false;
265 
266                         for h in H do
267                             if h[1] eq i and IsDivisibleBy(var*mon, h[2]) then
268                                 found := true;
269                                 break;
270                             end if;
271                         end for;
272 
273                         if not found then
274                             for t in [t : t in M[d-Degree(F[i])]`Signatures | t
    [1] lt i] do
275                                 index2 := t[1];
276                                 mon2 := t[2];
277                                 row2 := t[3];
278 
279                                 if LeadingMonomial(ExtractPolynomial(M[d-Degree
    (F[i])], row2)) eq var*mon then
280                                     found := true;
281                                     break;
282                                 end if;
283                             end for;
284                         end if;
285 
286                         if not found then
287                             AddPolynomial(~M[d], var*ExtractPolynomial(M[d-1],
    row), <index, var*mon, NumberOfRows(M[d])+1>);
288                         end if;
289                     end for;
290                 end for;
291             end if;
292         end for;
293 
294         T := Copy(M[d]);
295         Reduce(~M[d]);
296 
297         pols := [];
298         sigs := [];
```

```
        k := 1;

        for row := 1 to NumberOfRows(M[d]) do
            f := ExtractPolynomial(M[d], row);
            s := ExtractSignature(M[d], row);
            if f eq 0 then // found a syzygy
                Append(~H, <s[1], s[2]>);
                print "Row with signature ", s, "reduced to zero.";
            else
                Append(~pols, f);
                Append(~sigs, <s[1], s[2], k>);
                k +:= 1;

                if LeadingMonomial(ExtractPolynomial(T, row)) ne
    LeadingMonomial(ExtractPolynomial(M[d], row)) then
                    Append(~G, f);
                end if;
            end if;
        end for;

        M[d]`Polynomials := pols;
        M[d]`Signatures := sigs;

        delete T;

        G := Reduce(G);

        d +:= 1;
    until forall{m : m in MonomialsOfDegree(R, d) | exists{g : g in G |
    IsDivisibleBy(m, LeadingMonomial(g))}};

    return G;
end function;
```

A.2 The GVW algorithms

Listing A.3: The G2V algorithm

```
MyLeadingMonomial := function(f)
    if f eq 0 then
        return 0;
    else
        return LeadingMonomial(f);
    end if;
end function;

MyLeadingCoefficient := function(f)
    if f eq 0 then
        return 0;
    else
        return LeadingCoefficient(f);
    end if;
end function;

Swap := procedure(~a, ~b)
    t := a;
    a := b;
```

```
    b := t;
end procedure;

HeapParent := function(i)
    return Floor(i/2);
end function;

HeapLeft := function(i)
    return 2*i;
end function;

HeapRight := function(i)
    return 2*i + 1;
end function;

MinHeapify := procedure(~A, i)
    l := HeapLeft(i);
    r := HeapRight(i);

    if l le #A and A[l][1] lt A[i][1] then
        smallest := l;
    else
        smallest := i;
    end if;

    if r le #A and A[r][1] lt A[smallest][1] then
        smallest := r;
    end if;

    if smallest ne i then
        Swap(~A[i], ~A[smallest]);
        $$(~A, smallest);
    end if;
end procedure;

BuildMinHeap := procedure(~A)
    for i := Floor(#A/2) to 1 by -1 do
        MinHeapify(~A, i);
    end for;
end procedure;

HeapExtractMin := function(A)
    min := A[1];
    A[1] := A[#A];
    Undefine(~A, #A);
    MinHeapify(~A, 1);

    return A, min;
end function;

HeapDecreaseKey := procedure(~A, i, key)
    A[i] := key;
    while i gt 1 and A[HeapParent(i)][1] gt A[i][1] do
        Swap(~A[i], ~A[HeapParent(i)]);
        i := HeapParent(i);
    end while;
end procedure;
```

```
77
78  MinHeapInsert := procedure(~A, key)
79      HeapDecreaseKey(~A, #A+1, key);
80  end procedure;
81
82  MinHeapDelete := procedure(~A, i)
83      A[i] := A[#A];
84      Undefine(~A, #A);
85      MinHeapify(~A, i);
86  end procedure;
87
88  IsSuperTopReducible := function(U, V, p)
89      for i := 1 to #V do
90          if V[i] eq 0 and p[1] ne 0 and U[i] ne 0 and IsDivisibleBy(
        MyLeadingMonomial(p[1]), MyLeadingMonomial(U[i])) then
91              return true;
92          elif p[2] ne 0 and IsDivisibleBy(MyLeadingMonomial(p[2]),
        MyLeadingMonomial(V[i])) then
93              t := MyLeadingMonomial(p[2]) div MyLeadingMonomial(V[i]);
94              if MyLeadingMonomial(t*U[i]) eq MyLeadingMonomial(p[1]) and
        MyLeadingCoefficient(p[1])/MyLeadingCoefficient(U[i]) eq
        MyLeadingCoefficient(p[2])/MyLeadingCoefficient(V[i]) then
95                  return true;
96              end if;
97          end if;
98      end for;
99
100     return false;
101 end function;
102
103 FindReductor := function(U, V, p)
104     for i := 1 to #V do
105         if V[i] eq 0 and p[1] ne 0 and U[i] ne 0 and IsDivisibleBy(
        MyLeadingMonomial(p[1]), MyLeadingMonomial(U[i])) then
106             // super top reducible
107             continue;
108         elif p[2] ne 0 and IsDivisibleBy(MyLeadingMonomial(p[2]),
        MyLeadingMonomial(V[i])) then
109             t := MyLeadingMonomial(p[2]) div MyLeadingMonomial(V[i]);
110             c := MyLeadingCoefficient(p[2])/MyLeadingCoefficient(V[i]);
111             if MyLeadingMonomial(p[1]-c*t*U[i]) eq MyLeadingMonomial(p[1]) then
112                 return i;
113             end if;
114         end if;
115     end for;
116
117     return 0;
118 end function;
119
120 TopReduce := function(G, U, V, p)
121     i := FindReductor(U, V, p);
122     while i ne 0 do // 0 means p is not regular top reducible
123         // reduce p
124         t := MyLeadingMonomial(p[2]) div MyLeadingMonomial(V[i]);
125         c := MyLeadingCoefficient(p[2])/MyLeadingCoefficient(V[i]);
126
127         u := p[1]-c*t*U[i];
```

```
128         v := p[2]-c*t*V[i];
129
130         if MyLeadingMonomial(p[1]) eq t*MyLeadingMonomial(U[i]) then
131             u := u/(1-c);
132             v := v/(1-c);
133         end if;
134
135         v := NormalForm(v, G);
136
137         p := <u, v>;
138
139         i := FindReductor(U, V, p);
140     end while;
141
142     return p;
143 end function;
144
145 JPair := function(U, V, i, j)
146     t := LCM(MyLeadingMonomial(V[i]), MyLeadingMonomial(V[j]));
147     ti := t div MyLeadingMonomial(V[i]);
148     tj := t div MyLeadingMonomial(V[j]);
149
150     if ti*MyLeadingMonomial(U[i]) gt tj*MyLeadingMonomial(U[j]) then
151         return ti*MyLeadingMonomial(U[i]), i;
152     else
153         return tj*MyLeadingMonomial(U[j]), j;
154     end if;
155 end function;
156
157 IncrementBasis := function(G, g)
158     U := [Parent(g) ! 0 : i in [1..#G]];
159     V := G;
160     H := G;
161
162     v := NormalForm(g, G);
163     if v eq 0 then
164         Append(~H, Parent(g) ! 1);
165         return V, H;
166     else
167         Append(~U, 1);
168         Append(~V, v);
169     end if;
170
171     JP := [];
172     for i := 1 to #G do
173         t, j := JPair(U, V, i, #G+1);
174         if NormalForm(t, H) ne 0 then
175             found := false;
176             for k := 1 to #JP do
177                 if JP[k][1] eq t then
178                     if V[JP[k][2]] gt V[j] then
179                         JP[k] := <t, j>;
180                     end if;
181                     found := true;
182                 end if;
183             end for;
184
```

```
            if not found then
                Append(~JP, <t, j>);
            end if;
        end if;
    end for;

    BuildMinHeap(~JP);

    while not IsEmpty(JP) do
        JP, p := HeapExtractMin(JP);
        // while p equals min, do extractmin
        t := p[1];
        i := p[2];
        p := TopReduce(G, U, V, <(t div MyLeadingMonomial(U[i]))*U[i], (t div
    MyLeadingMonomial(U[i]))*V[i]>);

        if p[2] eq 0 then
            Append(~H, p[1]);
            L := [q : q in JP | IsDivisibleBy(q[1], MyLeadingMonomial(p[1]))];
            while not IsEmpty(L) do
                q := L[#L];
                Prune(~L);
                MinHeapDelete(~JP, Index(JP, q));
            end while;
        elif p[2] ne 0 and IsSuperTopReducible(U, V, p) then
            continue;
        else
            Append(~U, p[1]);
            Append(~V, p[2]);
            for i := 1 to #U-1 do
                t, j := JPair(U, V, #U, i);
                if NormalForm(t, H) ne 0 then
                    // for loop, if j is smaller than current, then update
                    // otherwise, insert

                    found := false;
                    for k := 1 to #JP do
                        if JP[k][1] eq t then
                            if V[JP[k][2]] gt V[j] then
                                JP[k] := <t, j>;
                            end if;
                            found := true;
                        end if;
                    end for;

                    if not found then
                        MinHeapInsert(~JP, <t, j>);
                    end if;
                end if;
            end for;
        end if;
    end while;

    return V, H;
end function;

MyGroebnerBasis := function(F)
```

```
    G := [F[1]];
    for i := 2 to #F do
        G := ReduceGroebnerBasis(IncrementBasis(G, F[i]));
    end for;

    return G;
end function;
```

Listing A.4: The GVW algorithm

```
MaxTerm := function(u, v)
    if u gt v then
        return u;
    else
        return v;
    end if;
end function;

Mylt := function(p, q)
    if Column(p) lt Column(q) then
        return true;
    elif Column(p) eq Column(q) and p[Column(p)] lt q[Column(q)] then
        return true;
    end if;

    return false;
end function;

Mygt := function(p, q)
    if Column(p) gt Column(q) then
        return true;
    elif Column(p) eq Column(q) and p[Column(p)] gt q[Column(q)] then
        return true;
    end if;

    return false;
end function;

Myeq := function(p, q)
    if Column(p) eq Column(q) and p[Column(p)] eq q[Column(q)] then
        return true;
    end if;

    return false;
end function;

MyLeadingMonomial  := function(f)
    if f eq 0 then
        return 0;
    else
        return LeadingMonomial(f);
    end if;
end function;

MyLeadingCoefficient  := function(f)
    if f eq 0 then
        return 0;
    else
```

```
        return LeadingCoefficient(f);
    end if;
end function;

Swap := procedure(~a, ~b)
    t := a;
    a := b;
    b := t;
end procedure;

HeapParent := function(i)
    return Floor(i/2);
end function;

HeapLeft := function(i)
    return 2*i;
end function;

HeapRight := function(i)
    return 2*i + 1;
end function;

MinHeapify := procedure(~A, i)
    l := HeapLeft(i);
    r := HeapRight(i);

    if l le #A and A[l][1] lt A[i][1] then
        smallest := l;
    else
        smallest := i;
    end if;

    if r le #A and A[r][1] lt A[smallest][1] then
        smallest := r;
    end if;

    if smallest ne i then
        Swap(~A[i], ~A[smallest]);
        $$(~A, smallest);
    end if;
end procedure;

BuildMinHeap := procedure(~A)
    for i := Floor(#A/2) to 1 by -1 do
        MinHeapify(~A, i);
    end for;
end procedure;

HeapExtractMin := function(A)
    min := A[1];
    A[1] := A[#A];
    Undefine(~A, #A);
    MinHeapify(~A, 1);

    return A, min;
end function;

```

```
HeapDecreaseKey := procedure(~A, i, key)
    A[i] := key;
    while i gt 1 and A[HeapParent(i)][1] gt A[i][1] do
        Swap(~A[i], ~A[HeapParent(i)]);
        i := HeapParent(i);
    end while;
end procedure;

MinHeapInsert := procedure(~A, key)
    HeapDecreaseKey(~A, #A+1, key);
end procedure;

MinHeapDelete := procedure(~A, i)
    A[i] := A[#A];
    Undefine(~A, #A);
    MinHeapify(~A, i);
end procedure;

ModuleDiv := function(p, q)
    return p[Column(p)] div q[Column(q)];
end function;

FindReductor := function(U, V, p)
    for i := 1 to #V do
        if V[i] ne 0 and p[2] ne 0 and IsDivisibleBy(MyLeadingMonomial(p[2]),
    MyLeadingMonomial(V[i])) then
            t := MyLeadingMonomial(p[2]) div MyLeadingMonomial(V[i]);
            c := MyLeadingCoefficient(p[2]) / MyLeadingCoefficient(V[i]);
            if Mylt(t*U[i], p[1]) or (Myeq(p[1], t*U[i]) and c ne 1) then
                return i;
            end if;
        end if;
    end for;

    return 0;
end function;

TopReduce := function(U, V, p)
    i := FindReductor(U, V, p);
    while i ne 0 do
        c := MyLeadingCoefficient(p[2]) / MyLeadingCoefficient(V[i]);
        t := MyLeadingMonomial(p[2]) div MyLeadingMonomial(V[i]);
        v := p[2]-c*t*V[i];

        if Myeq(p[1], t*U[i]) then
            v := v/(1-c);
        end if;

        p := <p[1], v>;
        i := FindReductor(U, V, p);
    end while;

    return p;
end function;

IsCoveredBy := function(p, U, V)
    for i := 1 to #U do
```

```
        if IsDivisibleBy(p[1], U[i]) then
            if ModuleDiv(p[1], U[i])*MyLeadingMonomial(V[i]) lt
    MyLeadingMonomial(p[2]) then
                return true;
            end if;
        end if;
    end for;

    return false;
end function;

IsDivisibleBySyzygy := function(p, H)
    for h in H do
        if IsDivisibleBy(p[1], h) then
            return true;
        end if;
    end for;

    return false;
end function;

JPair := function(U, V, i, j)
    t := LCM(MyLeadingMonomial(V[i]), MyLeadingMonomial(V[j]));
    ti := t div MyLeadingMonomial(V[i]);
    tj := t div MyLeadingMonomial(V[j]);

    assert ti*U[i] ne tj*U[j];

    if Mygt(ti*U[i], tj*U[j]) then
        return ti*U[i], i;
    else
        return tj*U[j], j;
    end if;
end function;

MyGroebnerBasis := function(F)
    M := EModule(Parent(F[1]), #F);
    U := [UnitVector(M, i) : i in [1..Degree(M)]];
    V := F;
    H := Setseq(Seqset([LeadingMonomial(F[j]*UnitVector(M, i)-F[i]*UnitVector(M
    , j)) : i, j in [1..#F] | i ne j]));

    JP := [];
    for i := 1 to #V do
        for j := 1 to i-1 do
                t, k := JPair(U, V, i, j);
                Include(~JP, <t,k>);
        end for;
    end for;

    BuildMinHeap(~JP);

    while not IsEmpty(JP) do
        JP, p := HeapExtractMin(JP);

        t := p[1];
        i := p[2];
```

```
        p := <t, ModuleDiv(t, U[i])*V[i]>;

        if IsDivisibleBySyzygy(p, H) or IsCoveredBy(p, U, V) then
            continue;
        end if;

        p := TopReduce(U, V, p);

        if p[2] eq 0 then
            if not IsDivisibleBySyzygy(p, H) then
                Include(~H, p[1]);
            end if;
        else
            Append(~U, p[1]);
            Append(~V, p[2]);

            for i := 1 to #U-1 do
                if MyLeadingMonomial(V[i])*p[1] ne MyLeadingMonomial(p[2])*U[i]
    then
                    Include(~H, MaxTerm(MyLeadingMonomial(V[i])*p[1],
    MyLeadingMonomial(p[2])*U[i]));
                    t, j := JPair(U, V, i, #U);
                    MinHeapInsert(~JP, <t, j>);
                end if;
            end for;
        end if;
    end while;

    return ReduceGroebnerBasis(V);
end function;
```

A.3 The quadratic systems method

Listing A.5: Functions related to the quadratic systems method

```
StarProduct := function(v, w)
    return Parent(v) ! [v[i]*w[i] : i in [1..NumberOfColumns(v)]];
end function;

Decode := function(C, y)
    if Length(C) ge #Field(C) then
        FE, _ := ext<Field(C) | Ceiling(Log(#Field(C), Length(C)))+1>;
        // There's a bug in Magma's ExtendField method
         C, _ := ExtendFieldCode(C, FE);
    end if;

    a := [PrimitiveElement(Field(C))^(i-1) : i in [1..Length(C)]];

    B := Matrix(Field(C), Length(C), Length(C), [<i, j, a[j]^(i-1)> : i, j in
    [1..Length(C)]]);
    Binv := B^(-1);
    mu := [[StarProduct(B[i], B[j])*Binv : j in [1..Length(C)]] : i in [1..
    Length(C)]];

    A := ParityCheckMatrix(C)*B^(-1);
    s := y*Transpose(ParityCheckMatrix(C));

```

```
21      G := [];
22      t := 1;
23      while true do
24          P<[X]> := PolynomialRing(Field(C), Length(C)+t, "grevlex");
25          J := [&+[A[j, l]*X[l] : l in [1..Length(C)]] - s[j] : j in [1..Length(C
        )-Dimension(C)]];
26          I := [&+[&+[mu[i,j,l]*X[l] : l in [1..Length(C)]]*X[Length(C)+j] : j in
         [1..t]]-&+[mu[i,t+1,l]*X[l] : l in [1..Length(C)]] : i in [1..Length(C)]];
27          sysQE := J cat I;
28
29          G := GroebnerBasis(sysQE);
30          if not 1 in G then
31              break;
32          end if;
33
34          t := t + 1;
35      end while;
36
37      try
38          u := VectorSpace(Field(C), Length(C)) ! [-MonomialCoefficient(G[i], 1)
        : i in [1..Length(C)]];
39      catch
40          error "Too many errors: ", e`Object;
41      end try;
42
43      e := u*Transpose(B^(-1));
44
45      return y-e;
46  end function;
47
48  MyMinimumDistance := function(C)
49      y := C ! 0;
50
51      if Length(C) ge #Field(C) then
52          FE, _ := ext<Field(C) | Ceiling(Log(#Field(C), Length(C)))+1>;
53          // There's a bug in Magma's ExtendField method
54           C, _ := ExtendFieldCode(C, FE);
55      end if;
56
57      a := [PrimitiveElement(Field(C))^(i-1) : i in [1..Length(C)]];
58
59      B := Matrix(Field(C), Length(C), Length(C), [<i, j, a[j]^(i-1)> : i, j in
        [1..Length(C)]]);
60      Binv := B^(-1);
61      mu := [[StarProduct(B[i], B[j])*Binv : j in [1..Length(C)]] : i in [1..
        Length(C)]];
62
63      A := ParityCheckMatrix(C)*B^(-1);
64      s := y*Transpose(ParityCheckMatrix(C));
65
66      G := [];
67      t := 1;
68      while true do
69          P<[X]> := PolynomialRing(Field(C), Length(C)+t, "grevlex");
70          J := [&+[A[j, l]*X[l] : l in [1..Length(C)]] - s[j] : j in [1..Length(C
        )-Dimension(C)]];
71          I := [&+[&+[mu[i,j,l]*X[l] : l in [1..Length(C)]]*X[Length(C)+j] : j in
```

```
    [1..t]]-&+[mu[i,t+1,l]*X[l] : l in [1..Length(C)]] : i in [1..Length(C)]];
72      sysQE := J cat I;
73
74      G := GroebnerBasis(sysQE);
75      u := [X[i] : i in [1..Length(C)]];
76      if not IsSubsequence(u, G) then
77          break;
78      end if;
79
80      t := t + 1;
81  end while;
82
83  return t;
84 end function;
```

A.4 A decoding heuristic

Listing A.6: A decoding heuristic

```
1  HeuristicDecode := function(P, G, t, r)
2      R<[X]> := P;
3      F := Field(Parent(r));
4      n := Degree(Parent(r));
5      a := PrimitiveElement(F);
6      A := [a^i : i in [0..#F-2]];
7
8      for i := 1 to #F-1 do
9          print "Trying with a = ", A[i];
10         w := A[i]*r;
11         w := &cat [Eltseq(b) : b in Eltseq(w)];
12
13         f := &*[X[i]^(Integers() ! w[i]) : i in [1..#w]];
14
15         f := NormalForm(f, G);
16         e := Exponents(f);
17         if Weight(VectorSpace(F, n) ! e) le t then
18             print "Found the right error vector: ", VectorSpace(F, n) ! e;
19             c := [w[i]-e[i] mod Characteristic(F) : i in [1..#w]];
20             c := Partition(c, Degree(F));
21             c := [Seqelt(b, F) : b in c];
22             c := VectorSpace(F, n) ! c;
23             c := A[i]^(-1)*c;
24             print "The codeword is: ";
25             return c;
26         else
27             print "Found the wrong error vector: ", VectorSpace(F, n) ! e;
28         end if;
29     end for;
30
31     return "fail";
32 end function;
```

A.5 Auxiliary functions

Listing A.7: Some auxiliary functions

```
1  MyDistinctDegreeFactorization := function(f)
```

```
    P<x> := Parent(f);
    q := #BaseRing(P);
    h := x mod f;
    G := [];

    while f ne 1 do
        h := Modexp(h, q, f);
        g := GCD(h-x, f);
        Append(~G, g);
        f := f div g;
        h := h mod f;
    end while;

    return G;
end function;

// Input: A squarefree monic polynomial f over F_q of degree n > 0 where q is an odd
// prime power, and a divisor d < n of n, so that all ireducible factors of f have
// degree d
// Output: A proper monic factor g over F_q of f, or "failure"
MyEqualDegreeSplitting := function(f, d)
    n := Degree(f);

    if n lt 1 then
        error "Runtime error 'MyEqualDegreeSplitting': first argument has degree less than 1";
    elif LeadingCoefficient(f) ne 1 then
        error "Runtime error 'MyEqualDegreeSplitting': first argument not monic";
    elif not IsSquarefree(f) then
        error "Runtime error 'MyEqualDegreeSplitting': first argument not squarefree";
    elif not IsDivisibleBy(n, d) then
        error "Runtime error 'MyEqualDegreeSplitting': second argument not a divisor of n";
    end if;

    P := Parent(f);
    F := BaseRing(P);
    q := #F;

    // pick a random polynomial of deg < n
    a := P ! [Random(F) : i in [1..n]];

    // a is the constant polynomial
    if a eq ConstantCoefficient(a) then
        return "failure";
    end if;

    g1 := GCD(a, f);
    // we're lucky and found a factor!
    if g1 ne 1 then
        return g1;
    end if;

```

```
    b := Modexp(a, (q^d-1) div 2, f);

    g2 := GCD(b-1, f);
    if g2 ne 1 and g2 ne f then
        return g2;
    else
        return "failure";
    end if;
end function;

// Input: A squarefree monic polynomial f over F_q of degree n > 0, for ann odd prime
// power q, and a divisor d of n, so that all irreducible factors of f have degree d.
// Output: The monic irreducible factors of f in F_q[x]
MyEqualDegreeFactorization := function(f, d)
    n := Degree(f);

    if n lt 1 then
        error "Runtime error 'MyEqualDegreeFactorization': first argument has degree less than 1";
    elif LeadingCoefficient(f) ne 1 then
        error "Runtime error 'MyEqualDegreeFactorization': first argument not monic";
    elif not IsSquarefree(f) then
        error "Runtime error 'MyEqualDegreeFactorization': first argument not squarefree";
    elif not IsDivisibleBy(n, d) then
        error "Runtime error 'MyEqualDegreeFactorization': second argument not a divisor of n";
    end if;

    // base case
    if n eq d then
        return [f];
    end if;

    // find a proper factor g of f
    repeat
        g := MyEqualDegreeSplitting(f, d);
    until Type(g) ne MonStgElt;

    // combine the results of the two recursive calls
    return $$(g, d) cat $$(f div g, d);
end function;

// Input: A nonconstant polynomial f in F_q[x], where q is an odd prime power.
// Output: The monic irreducible factors of f and their multiplicities.
MyFactorizationF := function(f)
    if f eq ConstantCoefficient(f) then
        error "Runtime error 'MyFactorizationF': argument is constant";
    end if;

    P<x> := Parent(f);
    q := #BaseRing(P);
    h := x mod f;
    v := f / LeadingCoefficient(f);
```

```
    i := 0;
    U := [];

    while v ne 1 do
        i := i+1;
        h := Modexp(h, q, f);
        g := GCD(h-x, v);

        if g ne 1 then
            G := MyEqualDegreeFactorization(g, i);

            for j := 1 to #G do
                e := 0;
                while IsDivisibleBy(v, G[j]) do
                    v := v div G[j];
                    e := e + 1;
                end while;
                Append(~U, <G[j], e>);
            end for;
        end if;
    end while;

    return U;
end function;

// Input: Nonzero polynomials f, g1,...,gs in F[x1,...,xn], where F is a field
// (and implicitly a monomial order on F).
// Output: Q = [q1,...,qs] and r such that f = q1g1 + ... + qsgs + r and no
    monomial in
// r is divisible by any of lt(g1),...,lt(gs).
MultivariateDivision := function(f, G)
    r := 0;
    p := f;
    s := #G;
    Q := [Parent(f) ! 0 : i in [1..s]];

    while p ne 0 do
        ltp := LeadingTerm(p);
        for i := 1 to s do
            ltg := LeadingTerm(G[i]);
            if IsDivisibleBy(ltp, ltg) then
                Q[i] := Q[i] + (ltp div ltg);
                p := p - (ltp div ltg)*G[i];
                break;
            end if;

            if i eq s then
                r := r + ltp;
                p := p - ltp;
            end if;
        end for;
    end while;

    return Q, r;
end function;

AffineHilbertSeries := function(I)
```

```
160     P<t> := PolynomialRing(Integers());
161     n := Rank(I);
162
163     G := GroebnerBasis(I);
164     LG := [LeadingMonomial(g) : g in G | g ne 0];
165
166     if #LG eq 0 then
167         return 1/(1-t)^(n+1);
168     end if;
169
170     if #LG eq 1 then
171         return (1-t^Degree(LG[1]))/(1-t)^(n+1);
172     end if;
173
174     J := Ideal([g : g in LG[2..#G]]);
175     Jbar := Ideal([LCM(LG[1], g) : g in LG[2..#G]]);
176
177     return (1-t^Degree(LG[1]))/(1-t)^(n+1) + $$(J) - $$(Jbar);
178 end function;
179
180 AffineHilbertPolynomial := function(I)
181     P<x> := PolynomialRing(Rationals());
182
183     s := AffineHilbertSeries(I);
184
185     k := Degree(Numerator(s))-Degree(Denominator(s))+1;
186
187     n := Degree(Denominator(s))-1;
188     a := Coefficients(Numerator(s));
189
190     j := Degree(Numerator(s));
191     if n eq 0 then
192         return -&+[a[i+1] : i in [0..j]], k;
193     end if;
194
195     p := &+[a[i+1]*(&*[(x+n-i+1-l)/l : l in [1..n]]) : i in [0..j]];
196
197     return p, k;
198 end function;
```