Bachelor Thesis

Daniel Rembold

Gröbner Fans for Linear Codes

August 2014

Advised by: Dipl.-Technomath. Natalia Dück
Supervised by: Prof. Dr. Dr. habil. Karl-Heinz Zimmermann

Technische Universität Hamburg-Harburg
Hamburg University of Technology
Institute of Computer Technology (E-13)
21071 Hamburg

Abstract

This work is about implementing a software, which enumerates the degree compatible as well as the whole Gröbner fan for linear codes.
The Gröbner fans from punctuated codes and the comparison between Gröbner fans are included too.

Zusammenfassung

In dieser Arbeit geht es um die Implementierung einer Software, die den grad-kompatiblen sowohl als auch den kompletten Gröbner-Fächer von linearen Codes berechnet.
Die Gröbner-Fächer von punktierten Codes sowie der Vergleich von Gröbner-Fächern können ebenfalls untersucht werden.

Ackknowledgements

I would like to thank Prof. Dr. Zimmermann for giving me the opportunity to write this thesis and the freedom to work on it my own way. Also many thanks to Dipl.- Technomath. Nathalia Dück for her valuable advices, her patience and that she took time for me if I had questions or problems. Last but not least thanks to my parents for their support throughout the years.

Danksagung

Ich möchte mich bei Prof. Dr. Zimmermann bedanken für die Freiheit beim Bearbeiten dieser Bachelorarbeit. Außerdem vielen Dank an Dipl.- Technomath. Nathalia Dück für ihre hilfreichen Anweisungen, ihre Geduld und dass sie sich immer Zeit für mich genommen hat, falls ich Fragen oder Probleme hatte. Zu guter Letzt bedanke ich mich bei meinen Eltern die mich von Anfang an unterstützt haben.

Statutory Declaration

I declare with my signature that I have authored this thesis independently and without any help. Content and passages from the literature, taken directly or indirectly are quoted. Furthermore, I assure that I have not used any other literature mentioned in the bibliography. This insurance refers to text content as well as all figures, sketches and tables. This work was not submitted nor published at any other auditing authority in identical or similar form.

Date: __________________________ Signature: __________________________

Eidesstaatliche Erklärung

Hiermit versichere ich an Eides statt und durch meine Unterschrift, dass die vorliegende Arbeit von mir selbstständig und ohne fremde Hilfe angefertigt worden ist. Inhalte und Passagen, die aus fremden Quellen stammen und direkt oder indirekt übernommen worden sind, wurden als solche kenntlich gemacht. Ferner versichere ich, dass ich keine andere, außer der im Literaturverzeichnis angegebenen Literatur verwendet habe. Diese Versicherung bezieht sich sowohl auf Textinhalte sowie alle enthaltenden Abbildungen, Skizzen und Tabellen. Die Arbeit wurde bisher keiner Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht in gleicher oder ähnlicher Form.

Datum: __________________________ Unterschrift: __________________________

Contents

List of Figures

List of Algorithms

List of Tables

1 Introduction

1.1 Motivation

The Gröbner fan of a polynomial ideal is a polyhedral complex consisting of cones in $\mathbb{R}^n_+$. Many applications of Gröbner bases rely on the computation of the Gröbner fan. Gröbner bases have the nice property that they solve the Ideal Membership Problem and can be useful to solve polynomial equations. Furthermore, the Gröbner fan can be used for a necessary condition for the code equivalence problem.
In many applications, the complete Gröbner fan is not needed, but only the degree compatible. Computing this certain part of the Gröbner fan could be dramatically faster than computing the whole fan.

1.2 Tasks

The purpose of this bachelor thesis is to implement an efficient software, which enumerates the degree compatible Gröbner fan of a linear code. But first, the concepts and algorithms have to be researched. The mathematical background is discussed and then the implementation and its results are explained.

1.3 Structure

Chapter 2 deals with the mathematical background. The first subsections explain polynomials, monomial ordering, ideals and the Ideal Membership Problem which is necessary for the Gröbner bases and Gröbner fans.
Chapter 3 bases on chapter 2. The algorithms to enumerate the Gröbner fan of a linear code are introduced which is the main task to implement in the software.
Chapter 4 presents the basic data structures of the implementation. Furthermore this section shows a tutorial how to use the software and some computational experience.
Chapter 5 gives an overview of the future possible extensions and improvements.
In Chapter 6 conclusions about the results are drawn.

2 Mathematical Background

In this chapter a mathematical basis is systematically approached to give the reader an understanding of Gröbner bases and Gröbner fans.
At first, monomials are revisited. The second section explains how monomials can be totally ordered. After that ideals are defined over polynomial rings and a summary on Gröbner bases and Gröbner fans for ideals is presented. Finally, the toric ideal is presented which gives the basis of the code ideals.

2.1 Monomials

In this section a brief explanation of polynomials is given.

Definition 2.1 (Monomial) *[1] A monomial is a product of variables over a finite field $\mathbb{K}$, denoted by $\mathbb{K}[X_1, X_2, \ldots, X_n]$ of the form $m = X_1^{u_1} X_2^{u_2} \cdots X_n^{u_n}$, where $u_i, 1 < i < n$ and $u \in \mathbb{N}_0$.*

The total **degree** of a monomial is $deg(m) = \sum_{i=1}^{n} u_i$.

Definition 2.2 (Polynomial) *[1] A polynomial f is a finite linear combination with coefficients $c_u \in \mathbb{K}$ multiplied with monomials, such that*

$$f = \sum_u c_u X^u.$$

If $c_u \neq 0$ then $c_u x_u$ is a term of f.

2.2 Monomial Order

It is necessary to rearrange a polynomial with respect to a monomial order. That forms the foundation for dividing polynomials in the finite field $\mathbb{K}[X_1, X_2, \ldots, X_n]$ and solving the Ideal Membership Problem.
The Ideal Membership Problem describes if a polynomial lies in an ideal I, in other words, if a polynomial divided by an Ideal I has a zero remainder, the polynomial lies in I. Ideals will be defined in section 2.3.

Definition 2.3 (Monomial Ordering) *[3] A monomial order is a relation $>$ on the set of all monomials in $\mathbb{K}[x]$. Let m_1, m_2 and m_3 be monomials:*

- *for any pair of monomials m_1, m_2, either $m_1 > m_2$ or $m_2 > m_1$ or $m_1 = m_2$*
- *if $m_1 > m_2$ and $m_2 > m_3$ then $m_1 > m_3$*
- *$m_1 > 1$ for any monomial $m_1 \neq 1$*
- *if $m_1 > m_2$ then $m \cdot m_1 > m \cdot m_2$ for any monomial m.*

Two commonly used monomial orders are the following. Let u and v be elements of $\mathbb{N}_0^n$.
Lexicographic Order [3] $u >_{lex} v$ if in $u - v$ the left most non-zero entry is positive. This can be written as $X^u >_{lex} X^v$ if $u >_{lex} v$.

Graded Lex Order [3] $u >_{grlex} v$ if $deg(u) > deg(v)$ or if $deg(u) = deg(v)$ and $u >_{lex} v$.

Example 1 Let $m_1 = x^2y^4z^3$ and $m_2 = x^1y^1z^4 \in \mathbb{K}[x, y, z]$. The monomials can also be written as $m_1 = X^{(2\ 4\ 3)}$ and $m_2 = X^{(1\ 1\ 4)}$. Thus $m_1 >_{lex} m_2$ because the left most non-zero entry of $(2\ 4\ 3) - (1\ 1\ 4)$ is positive.

The total degree of m_1 is 9 and $deg(m_2) = 6$. Hence, $deg(m_1) > deg(m_2)$ so that $m_1 >_{grlex} m_2$.

$\diamondsuit$

Weight vectors
In order to compare monomials with a generic vector $c = (c_1, \ldots, c_n) \in \mathbb{R}^n_+$, the dot product with the exponent vector has to be taken. The highest result is the leading term. If a tie occurs, some other fixed monomial order has to be used. Note that the standard monomial orders can be expressed as weight vectors. The lexicographic order needs for instance the first unit vector, if a tie occurs the second unit vector and so on.

Leading term

Given a monomial order $>$, each non-zero polynomial $f \in \mathbb{K}[x]$ has a unique leading term, denoted by $\mathrm{LT}(f)$, given by the largest involved term with respect to the monomial order.
If $\mathrm{LT}(f) = cX^u$, where $c \in \mathbb{K}$, then c is the leading coefficient of f and X^u is the leading monomial (LM) or the initial monomial of f [1].

Example 2 Let $f = 3x^2y^5z^3 + x^4 - 2x^3y^4 + 12x^2z^2$
With respect to lex order : $f = \underline{x^4} - 2x^3y^4 + 3x^2y^5z^3 + 12x^2z^2$
with respect to grlex order : $f = \underline{3x^2y^5z^3} - 2x^3y^4 + x^4 + 12x^2z^2$
with respect to the weight vector $(3, 2, 1)$: $f = \underline{3x^2y^5z^3} - 2x^3y^4 + x^4 + 12x^2z^2$
The underlined terms are the leading terms with the respect to the monomial order.

$\diamondsuit$

2.3 Ideals

Definition 2.4 (Ideal) *[1] An ideal I is a collection of polynomials $f_1, \ldots, f_s \in \mathbb{K}[X_1, \ldots, X_n]$ and polynomials which can be built from these by addition and multiplication with arbitrary polynomials.*

This is called an ideal generated by $f_1, \ldots, f_s$:

$$\langle f_1, \ldots, f_s \rangle = \left\{ \sum_{i=1}^{s} h_i f_i \mid h_1, \ldots, h_s \in \mathbb{K}[X_1, \ldots, X_n] \right\}.$$

An ideal satisfies [3]:

- $0 \in I = \langle f_1, \cdots, f_s \rangle$
- If $f, g \in \langle f_1, \cdots, f_s \rangle$, then $f + g \in \langle f_1, \cdots, f_s \rangle$
- If $f \in \langle f_1, \cdots, f_s \rangle$ and $h \in \langle f_1, \cdots, f_s \rangle$, then $f \cdot h \in \langle f_1, \cdots, f_s \rangle$.

Example 3 Let $I = \langle f_1, f_2 \rangle = \langle x^2+y, x+y+1 \rangle$ and $f = yx^2+y^2+x^2+xy+x$. Since $f = y \cdot f_1 + x \cdot f_2,\ f \in I$.

$\Diamond$

Definition 2.5 (Leading Ideal) *[2] The leading ideal of I with respect to a monomial order $>$ on $\mathbb{K}[X_1, \ldots, X_n]$ is the monomial ideal*

$$lt_>(I) = \langle lt_>(f) : f \in I \rangle.$$

The monomial $lt_>(f)$ means the leading term of f with respect to a monomial order $>$. In other words, the leading ideal of I is the ideal generated by its leading terms.

2.4 Division Algorithm

The Ideal Membership Problem is easy to solve in a polynomial ring with one variable. It is only necessary to apply the polynomial division, which means dividing the polynomial by the ideal and to check if the remainder is zero. If the result has zero remainder, the polynomial p lies in the ideal I. But in a ring with several variables like $\mathbb{K}[X_1,\ X_2, \ldots,\ X_n]$, the usual division algorithm will not work. A generalized algorithm is needed.

The goal is to divide $g \in \mathbb{K}[X_1, \ldots, X_n]$ by $f_1, \ldots,\ f_s \in \mathbb{K}[X_1, \ldots, X_n]$, so g can be expressed in the form

$$g = a_1 f_1 + \ldots + a_s f_s + r,$$

where the $a_1 f_1 + \ldots + a_s f_s$ and $r \in \mathbb{K}[X_1, \ldots, X_n]$ [4]. The remainder r is zero or r is a linear combination with the coefficients in $\mathbb{K}$ and none of them are divisible by any of $\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_s)$. Furthermore, if $a_i f_i \neq 0$, then $deg(g) \geq deg(a_i f_i)$. Algorithm 1 shows how to divide polynomials in a commutative ring with more than one variable.

Algorithm 1 Division Algorithm [1]

Require: Basis $\langle f_1, \ldots, f_m \rangle$ of nonzero polynomials
Ensure: $r = 0$ **or** none of the terms in r are divisible by $LT_{\leq}(f_1), \ldots, LT_{\leq}(f_m)$

$h_1 \leftarrow 0, \ldots, h_m \leftarrow 0;\ r \leftarrow 0;\ s \leftarrow f$
while $s \neq 0$ **do**
    $i \leftarrow 1$
    division_occured $\leftarrow$ false
    while $i \leq m$ **and** division_occured = false **do**
        if $LT(f[i])$ divides $LT(s)$ **then**
            $s \leftarrow s - (LT(s)/LT(f[i])) \cdot f_i$
            $h_i \leftarrow h_i + LT(s)/LT(f_i)$
            division_occured = true
        else
            $i \leftarrow i + 1$
        end if
    end while
    if division_occured = false **then**
        $r \leftarrow r + \mathrm{LT}(s)$
        $S \leftarrow s - \mathrm{LT}(s)$
    end if
end while

Example 4 Consider the ideal $I = \langle f_1, f_2 \rangle = \langle xy^2 + z, y^2 - 1 \rangle$ and the polynomial $f = x^3y^2 + x^2z$. First, with respect to the lex-order, applying algorithm 1 gives the expression : $f = x^2(xy^2 + z) + 0(y^2 - 1) + 0$.
But the division with f and $I = \langle f_2, f_1 \rangle$ gives the expression
$f = x^3(y^2 - 1) + x^2(xy^2 + z) - x^3y^2 + x^3$.

$\diamondsuit$

Example 4 shows that is still possible to obtain a non-zero remainder even if $f \in \langle f_1, f_2 \rangle$. That means r = 0 is a necessary condition for the ideal membership but not a sufficient condition.

2.5 Gröbner basis

The solution of the Ideal Membership Problem needs a certain generating set for an ideal I. It would be helpful if the remainder r on division is uniquely determined and the condition $r = 0$ is equivalent to the ideal membership.

Definition 2.6 (Gröbner basis) *[1] Let $\leq$ be a monomial order on $\mathbb{K}[X_1, \ldots, X_n]$ and let I be an ideal on $\mathbb{K}[X_1, \ldots, X_n]$. A Gröbner basis G for I (with respect to $\leq$) is a finite set of polynomials $G = \{f_1, \ldots, f_m\}$ in I with the property that for every nonzero $f \in I$, $\mathrm{LT}_{\leq}(f)$ is divisible by $\mathrm{LT}_{\leq}(f_i)$ for some $1 \leq i \leq m$.*

A Gröbner basis has the beneficial property that the remainder r of f divided by the elements of a Gröbner basis G are uniquely determined and independent of the order of the elements in G. Also every ideal in $\mathbb{K}[X_1, \ldots, X_n]$ has a Gröbner basis with respect to any monomial order [1].

In order to obtain a Gröbner basis of an arbitrary basis $\{f_1, \ldots, f_n\}$ with an arbitrary monomial order $>$ of an ideal I, an algorithm is needed. This algorithm is called Buchberger-Algorithm and is defined at algorithm 2. The idea is to build every possible S-Polynomial of (f_i, f_j) for every $1 \leq i \neq j \leq n$ and every nonzero result is added to the basis until every S-Polynomial of (f_i, f_j) vanishes.

Let the polynomials $f, g \in \mathbb{K}[X_1, \ldots, X_n]$ and $\mathrm{LT}_{\leq}(f) = cX^{\alpha}$, $\mathrm{LT}_{\leq}(g) = dX^{\beta}$ and $\mathrm{LCM}\left(X^{\alpha}, X^{\beta}\right)$ be the least common multiple between X^{α} and X^{β}.

Definition 2.7 (S-Polynomial) *[1] The S-Polynomial of f and g is the polynomial*

$$S(f, g) = \frac{\mathrm{LCM}\left(X^{\alpha}, X^{\beta}\right)}{\mathrm{LT}_{\leq}(f)} \cdot f - \frac{\mathrm{LCM}\left(X^{\alpha}, X^{\beta}\right)}{\mathrm{LT}_{\leq}(g)} \cdot g.$$

Example 5 Consider the ideal in the polynomial ring $\mathbb{K}[x, y, z]$ with the basis $\{f, g\} = \{xy^2 - xz + y, xy - z^2\}$ with respect to the lexicographic order. Forming the S-Polynomial leads to:

$$
\begin{aligned}
S(f,\ g) &= \frac{\mathrm{LCM}(xy^2, xy)}{xy^2} \cdot \left(xy^2 - xz + y\right) - \frac{\mathrm{LCM}(xy^2,\ xy)}{xy} \cdot (xy - z) \\
&= \frac{xy^2}{xy^2} \cdot \left(xy^2 - xz + y\right) - \frac{xy^2}{xy} \cdot (xy - z) \\
&= -xz - yz + y.
\end{aligned}
$$

◊

The S-Polynomial is not zero and is not divisible by the leading terms of f or g. That means the basis given in the example is not a Gröbner basis. This can be deduced by the following assertion.

Definition 2.8 (Buchberger Criterion) *[1] A finite set $G = \{f_1, \ldots, f_m\}$ of polynomials in $\mathbb{K}[X_1, \ldots, X_n]$ is a Gröbner basis of an ideal $I = \langle f_1, \ldots, f_m \rangle$ if and only $S(f_i, f_j) = 0$, $\forall\ 1 \leq i, j \leq n, i \neq j$.*

Algorithm 2 Buchberger Algorithm [1]

Require: Basis $F = (f_1, \ldots, f_m)$
Ensure: Gröbner basis G for $I = \langle f_1, \ldots, f_m \rangle$ with $F \subseteq G$

$G \leftarrow F$
repeat
  $G' \leftarrow G$
  for each pair f_i and f_j in G , $i \neq j$ **do**
    $S \leftarrow S(f_i, f_j)^{G'}$ ▷ S-Polynomial with the basis of G'
    if $G \neq 0$ **then**
      $G \leftarrow G \cup \{S\}$
    end if
  end for
until $G = G'$

This algorithm is correct and terminates [1].

However, a Gröbner basis is not unique. An arbitrary polynomial can be added to a Gröbner basis and will remain a Gröbner basis. Fortunately, each nonzero ideal in $\mathbb{K}[X_1, \ldots, X_n]$ has a unique *reduced* Gröbner basis with respect to a fixed monomial order.

Definition 2.9 (Reduced Gröbner basis) *[1] A Gröbner basis $G = \{f_1, \ldots, f_m\}$ in $\mathbb{K}[X_1, \ldots, X_n]$ is reduced if the polynomials $f_1, \ldots, f_m$ are monic and no term f_i is divisible by $\mathrm{LT}_{\leq}(f_j)$ for any pair $i \neq j$, where $\leq$ is a monomial order.*

2.6 Gröbner fans

Gröbner bases for a fixed ideal I with different monomial orders can look very different and have different properties. The difference can be in the number of elements in the Gröbner basis, the size or the degree of the elements. So it will be helpful if all possible Gröbner bases of a fixed ideal can be collected together.
Even though there are infinite monomial orders for an ideal, the amount of reduced Gröber bases are finite [4].

Definition 2.10 (Gröbner fan) *[4] A Gröbner fan of an ideal I consists of finitely many closed convex polyhedral cones with vertices at the origin, such that :*

- *A face of a cone σ is $\sigma \cap \{l = 0\}$, where $l = 0$ is a nontrivial linear equation such that $l \geq 0$ on σ. Any face of a cone in the fan is also in the fan.*
- *The intersection of two cones in the fan is a face of each.*

In order to construct a Gröbner fan to a given ideal, consider the marked Gröbner basis $G = \{g_1, \ldots, g_t\}$ of the ideal I. A marked Gröbner basis is a Gröbner basis where each $g \in G$ has an identified leading term, such that G

is a reduced Gröbner basis with respect to some monomial order > selecting those terms. Informally, where all leading terms in G are marked.

The elements of G, g_i , can be written as

$$g_i = x^{\alpha(i)} + \sum_{\beta} c_{i,\beta} \cdot x^{\beta},$$

where $x^{\alpha(i)}$ is the leading term and $x^{\alpha(i)} > x^{\beta}$, with respect to a monomial order, whenever $c_{i,\beta} \neq 0$.
If a weight vector $\mathbf{w}$ satisfies the inequality $\alpha(i) * \mathbf{w} \geq \beta * \mathbf{w}$, the vector selects the correct leading term in g_i as the term with the highest weight.

So, the cone of a Gröbner basis can be written as : [4]

$$C_G = \left\{ \mathbf{w} \in (\mathbb{R}^n)^+ : \alpha(i) \cdot \mathbf{w} \geq \beta \cdot \mathbf{w} \quad \text{whenever } c_{i,\beta} \neq 0 \right\}.$$

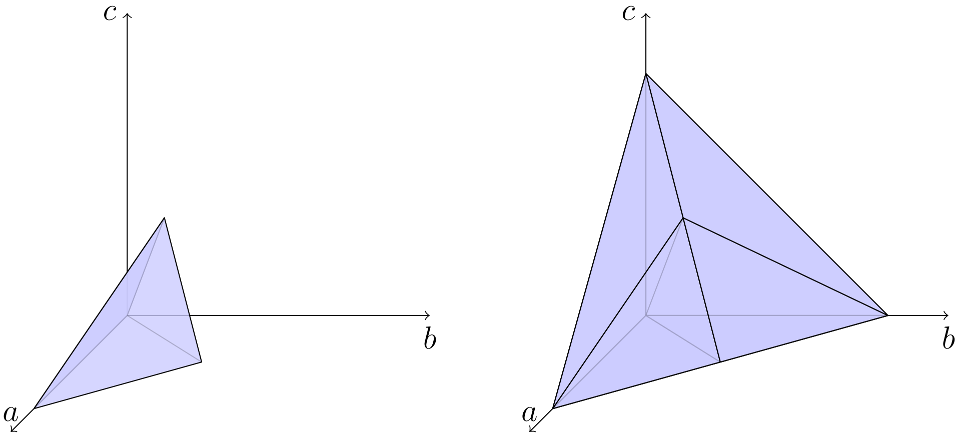
Example 6 Consider the ideal with $I = \langle x^2 - z, y - x \rangle \in \mathbb{Q}[x, y, z]$. Note that the ring has 3 variables so that the Gröbner fan can be plotted in the positive orthant $\mathbb{R}^3_+$.
The marked Gröbner basis with respect to the *lex* order with $x > y > z$ is

$$G^{(1)} = \left\{ \underline{y^2} - z, \underline{x} - y \right\}.$$

The leading terms are underlined. Let $\mathbf{w} = (a, b, c) \in \mathbb{R}^3_+$. Then $\mathbf{w}$ is in the cone $C_{G^{(1)}}$ if and only if the inequalities defined as above are satisfied.

- $(0, 2, 0) \cdot (a, b, c) \geq (0, 0, 1) \cdot (a, b, c)$ **or** $2b \geq c$
- $(1, 0, 0) \cdot (a, b, c) \geq (0, 1, 0) \cdot (a, b, c)$ **or** $a \geq b$

This is the first cone of the Gröbner fan and can be drawn in the positive orthant. It is sliced of the plane of $a + b + c = d$, $d \in \mathbb{R}_+$ for a better clarity.



(a) Gröbner cone for the first basis (b) Complete Gröbner fan

Figure 1: Gröbner fans for example 6

Figure 1a shows that the Gröbner fan is not complete, since the cone does not cover the whole positive orthant. In this example, the other reduced Gröbner basis can be obtained by applying the Buchberger-Algorithm with common monomial orders to the ideal I. If the computed cones still do not cover the whole positive orthant, then further computations with weight vectors are necessary.

$\diamond$

Example 6 illustrates clearly that an arbitrary non-negative weight vector can be selected and if the vector lies in a certain cone, the corresponding Gröbner basis will match with respect to this weight vector.

This strategy is reasonable for small examples like above. In general, the whole Gröbner fan can be computed with the Gröbner walk [4].
An inexpensive way to obtain all reduced Gröbner bases of a special ideal, the code ideal, will be explained in section 3.1.

2.7 Toric Ideals

This work is focused on code ideals, so it is useful to define toric ideals first. Given a matrix $A = [a_1, \ldots, a_n] \in \mathbb{Z}^{d \times n}$ and $u \in \mathbb{Z}^n$, which can be decomposed in u^+ and u^-, where u^+ and u^- have non-negative coefficients and disjoint support, the toric ideals can be defined as follows.

Definition 2.11 (Toric Ideal) *[5] A toric ideal I_A is defined as*

$$\boldsymbol{I}_A = \langle \boldsymbol{x}^{u^+} - \boldsymbol{x}^{u^-} \mid u \in ker\,(A) \rangle.$$

The toric ideal can also be expressed as

$$\mathbf{I}_A = \langle \mathbf{x}^u - \mathbf{x}^v \mid Au = Av,\ u, v \in \mathbb{N}_0^n \rangle.$$

3 Enumerating Gröbner fans and Code Ideals

Now that the mathematical background is given, the specific parts which are needed to implement the software and the connection between linear codes and Gröbner bases will be presented. The first section deals with enumerating all Gröbner bases with the help of breadth-first and reverse search followed by the degree compatible Gröbner bases. After that, linear codes will be presented and to connect these two topics together, code ideals will be defined.

3.1 Enumerating Gröbner fans

In this section, two algorithms will be explained with the purpose to enumerate the Gröbner fan. To compute all Gröbner bases from a toric ideal I_A , it is necessary to search the *edge graph* of a Gröbner fan.
The Gröbner bases are the vertices of the *edge graph*. Two reduced Gröbner bases with respect to a term order covered by the generic weight vectors c_1, c_2 are said to be adjacent in the *edge graph* if the two Gröbner cones share a common *facet*. Given a reduced Gröbner basis with respect to the weight vector c, *facet binomials* can be defined as follows.

Definition 3.1 (Facet Binomial) *[2] The binomial $x^{\alpha_k} - x^{\beta_k} \in \mathcal{G}_c$, $\mathcal{G}_c$ is a reduced Gröbner basis with respect to the weight vector c, is a facet binomial of $\mathcal{G}_c$ if and only if there exists a vector $u \in \mathbb{R}^n$ which satisfies :*

- $\{\alpha_i \cdot u > \beta_i \cdot u : i = 1, \ldots, t, i \neq k\}$
- $\{\beta_k \cdot u > \alpha_k \cdot u\}$.

Computing the facet binomials of a reduced Gröbner basis $\mathcal{G}$ can be computationally expensive, because it is needed to solve as many linear programs as the cardinality of $\mathcal{G}$. Algorithm 3 for finding the facets can defined be as follows.

Algorithm 3 Finding the facets of a reduced Gröbner basis of I_A [2]

Input: Reduced Gröbner basis $\mathcal{G} = \{\underline{x}^{a_i} - x^{b_i} : i = 1, \dots, t\}$ of I_A
Output: The facet binomials of $\mathcal{G}$

1: Facets := $\emptyset$
2: **for each** binomial $\underline{x}^{a_i} - x^{b_i}$ in $\mathcal{G}$ **do**
3:     **if** $a_i - b_i \notin$ cone generated by $\{a_j - b_j : \underline{x}^{a_j} - x^{b_j} \in \mathcal{G}, i \neq k\}$ **then**
4:         Facets := Facets $\cup \{x^{a_i} - x^{b_i}\}$
5:     **end if**
6: **end for**
7: **return** Facets

Example 7 Consider the reduced Gröbner basis
$\mathcal{G} = \{x_1 - x_5,\ x_2^2 - 1,\ x_2x_4 - x_5x_6,\ x_2x_6 - x_4x_5,\ x_3 - x_5,\ x_4^2 - 1,\ x_4x_5x_6 - x_2,\ x_5^2 - 1,\ x_6^2 - 1\}$ and it shall be determined if the term $x_2x_4 - x_5x_6$ is a facet or not. With linear programming, it must be checked if the vector $b = (0, 1, 0, 1, -1, -1)^T$ can be expressed by the matrix multiplication $A \cdot x = b$, where

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 2 & 1 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 2 \end{pmatrix}.$$

Now the linear program can be set up to

$$\begin{array}{rrcl} & Ax & = & b \\ \text{subject to} & x & \geq & 0 \end{array}$$

$\Diamond$

Another way to find the facets without linear programming is possible with the following property of the leading ideals.

Let $\mathcal{G}_c$ be a reduced Gröbner basis with respect to a generic weight vector c. Then $x^a - x^b$ is a facet binomial of $\mathcal{G}_c$ only if $lt_c(I_a)$ (see definition 2.5) is the leading ideal of

$$W_{a-b} = \langle x^a - x^b \rangle \ + \ \langle x^c \mid x^c \text{ is a minimal generator of } lt_c(I_a), x^c \neq x^a \rangle$$

[2]. With the help of this property it is possible to define an algorithm to find a superset of facet binomials of a reduced Gröbner basis.

Algorithm 4 Finding a superset of the facet binomials of a reduced Gröbner basis of I_A [2]

Input: Reduced Gröbner basis $\mathcal{G}_c$ of I_A
Output: A superset SS of the facet binomial $\mathcal{G}_c$

```
SS := ∅
for each x^a − x^b ∈ G_c do
    W_{a−b} := ⟨x^a − x^b⟩ + ⟨x^c⟩                         ▷ x^c is defined as above
    if lt_c(I_A) is the leading ideal of W_{a−b} with respect to x_a >_c x_b then
        SS := SS ∪ {x^a − x^b}
    end if
end for
return SS
```

Example 6 shows that the facet binomials of a Gröbner cone determine the borders to other Gröber cones. In order to traverse from a Gröber base $\mathcal{G}_c$ to a neighboured Gröbner base $\mathcal{G}_{c'}$ with the certain facet $x^{\alpha} - x^{\beta}$, a procedure making a local change from $\mathcal{G}_c$ to $\mathcal{G}_{c'}$ is required. This procedure is called *flip* and is presented in algorithm 5.

Algorithm 5 Local change of reduced Gröbner bases in I_A [2]

Input: Reduced Gröbner basis $\mathcal{G}$ of I_A **and**
A prescribed facet binomial $\underline{x_i^a} - x_i^b \in \mathcal{G}$

Output: The reduced Gröbner basis is adjacent to $\mathcal{G}$ in which $\underline{x_i^b} - x_i^a$ is a facet binomial.

1: Old := $\{\underline{x_i^a} - x_i^b\} \cup \{\underline{x_j^a} : \underline{x_j^a} - x_j^b \in \mathcal{G}, j \neq i\}$ ▷ Prescribed Facet and leading terms only in Old
2: Temp := $\{\underline{x_i^b} - x_i^a\} \cup \{\underline{x_j^a} : x_j^a \in Old\}$ ▷ Flipping the facet Binomial
3: New := Reduced Gröbner basis with respect to the new marking ▷ See algorithm 2.
4: $\mathcal{G}' = \left\{\underline{x_i^b} - x_i^a\right\}$
5: **for each** monomial h in New **do**
6: Reduce h with $\mathcal{G}$ to obtain the monomial h'.
7: Add $h - h'$ to $\mathcal{G}'$ with h marked as the leading term.
8: **end for**
9: Auto-reduce $\mathcal{G}'$ to get $\mathcal{G}_{new}$ ▷ no term shall be divisible by a leading term

This algorithm is correct and can terminate [2]. The main advantage of the algorithm is that no weight vectors must be stored or computed. Weight vectors are carried implicitly and that is possible due to the binomial structure that this subroutine generates for every Gröbner basis. For this work flip($x_i^a - x_i^b,\ \mathcal{G}$) means that algorithm 5 is applied with the needed input.

Example 8 Consider the reduced Gröbner basis with respect to a monomial order $\succ$: $\mathcal{G} = \{x_6^2 - 1,\ x_5^2 - 1,\ x_4x_5x_6 - x_2,\ x_4^2 - 1,\ x_3 - x_5,\ x_2x_6 - x_4x_5,\ x_2x_5 - x_4x_6,\ x_2x_4 - x_5x_6,\ x_2^2 - 1,\ x_1 - x_5\}$.

Applying the procedure flip($x_3 - x_5, \mathcal{G}$) leads to:

Old := $\{x_3 - x_5,\ x_6^2,\ x_5^2,\ x_4x_5x_6,\ x_4^2,\ x_2x_6,\ x_2x_5,\ x_2x_4,\ x_2^2,\ x_1\}$
Temp := $\{x_5 - x_3,\ x_6^2,\ x_5^2,\ x_4x_5x_6,\ x_4^2,\ x_2x_6,\ x_2x_5,\ x_2x_4,\ x_2^2,\ x_1\}$

Now the new Gröbner basis has to be calculated (see line 3 at algorithm 5), but first it useful to know that all pairs of binomials which have the greatest common divisor of 1 will be auto-reduced to zero. In other words, it is only necessary to form the S-Polynomials of binomials that are not relatively prime.

$$
\begin{aligned}
S(x_5 - x_3, x_5^2) &= x_3x_5 \\
S(x_5 - x_3, x_4x_5x_6) &= x_3x_4x_5 \\
S(x_5 - x_3, x_2x_5) &= x_2x_3 \\
S(x_5 - x_3, x_3x_5) &= x_3^2
\end{aligned}
$$

Now the new Gröbner basis $\mathcal{G}'$ shall be filled with all monomials from $\mathcal{G}$. The underlined terms are the terms reduced by $\mathcal{G}$ and will be the new non-leading terms.

$$
\begin{aligned}
x_3x_5 &= x_5(x_3 - x_5) + 1(x_5^2 - 1) && +\underline{1} \\
x_3x_4x_6 &= x_4x_6(x_3 - x_5) + 1(x_4x_5x_6 - x_2) && +\underline{x_2} \\
x_2x_3 &= x_2(x_3 - x_5) + 1(x_2x_5 - x_4x_6) && +\underline{x_4x_6} \\
x_3^2 &= x_3(x_3 - x_5) + x_5(x_3x_5) + 1(x_5^2 - 1) && +\underline{1} \\
x_1 &= (x_1 - x_5) + (x_5 - x_3) && +\underline{x_3} \\
x_2x_6 &= (x_2x_6 - x_4x_5) + x_4(x_5 - x_3) && +\underline{x_3x_4} \\
x_2x_4 &= (x_2x_4 - x_5x_6) + x_6(x_5 - x_3) && +\underline{x_3x_6}
\end{aligned}
$$

The other monomials of $\mathcal{G}$ are not notated here because they did not get a new non-leading term.

This results to the Gröbner base :
$\mathcal{G}' = \{x_5 - x_3,\ x_6^2 - 1,\ x_5^2 - 1,\ x_4x_5x_6 - x_2,\ x_4^2 - 1,\ x_2x_6 - x_3x_4,\ x_2x_5 - x_4x_6,\ x_2x_4 - x_3x_6,\ x_2^2 - 1, x_1 - x_3,\ x_3x_5 - 1,\ x_3x_4x_6 - x_2,\ x_2x_3 - x_4x_6,\ x_3^2 - 1\}$.
This Gröbner basis is not reduced yet. After canceling out all binomials whose terms are divisible by some other leading terms the new reduced Gröbner basis is $\mathcal{G}_{new} = \{x_5 - x_3,\ x_6^2 - 1,\ x_4^2 - 1,\ x_2x_6 - x_3x_4,\ x_2x_4 - x_3x_6,\ x_2^2 - 1,\ x_1 - x_3,\ x_3x_4x_6 - x_2,\ x_2x_3 - x_4x_6,\ x_3^2 - 1\}$.

$\Diamond$

3.1.1 Breadth first search

In this section an algorithm to enumerate the edge graph of a Gröbner fan via breath-first search and its drawbacks are presented.

Algorithm 6 Enumerating the edge graph of the Gröbner fan via breath-first search [2]

Input: Any reduced Gröbner basis $\mathcal{G}_0$ of I_A
Output: All reduced Gröbner bases of I_A, (all vertices of the edge graph)

Todo := $[\mathcal{G}_0]$
Verts := []
while Todo $\neq \emptyset$ **do**
  $\mathcal{G}$:= first element in (Todo)
  Remove $\mathcal{G}$ from Todo
  add $\mathcal{G}$ to Verts
  determine list L of facet binomials of $\mathcal{G}$ ▷ With algorithm 3 or 4
  for each $x^\alpha - x^\beta \in$ L **do**
    $\mathcal{G}' =$ flip$(\mathcal{G}, x^\alpha - x^\beta)$ ▷ Applying algorithm 5
    if $\mathcal{G}' \notin$ Todo $\cup$ Verts **then**
      add $\mathcal{G}'$ to Todo
    end if
  end for
end while
return Verts

This algorithm is intuitive but has the drawback that every vertex of the edge graph must be stored and every vertex must be checked against all other vertices if it is a new vertex or not. The more vertices the edge graph has, the more expensive the calculation will be. Also the need of memory will arise if a Gröbner fan has a lot of cones.

3.1.2 Reverse search tree

The disadvantages can be canceled out with a memoryless algorithm, which runs linear depending on the size of output. The idea is to enumerate the edge graph with a depth-first reverse search. The result will be a directed subgraph of the edge graph, called reverse search tree $T_{\succ}(I_A)$.

Definition 3.2 (Mismarked Polynomial) *[2] A polynomial f that has been marked with respect to the monomial order $>$ is mismarked with respect to the monomial order $>'$.*

Example 9 Consider the reduced Gröbner base
$\mathcal{G} = \{x^2y - z, y^2 - xz, zy - xy^2z\}$ with a certain monomial order $>$, which is not the lexicograpic order. Then, the second and last term are clearly mismarked with respect to $>_{lex}$.
Applying the flip-procedure $(y^2 - xz,\ \mathcal{G})$ leads to the Gröbner base
$\mathcal{G} = \{x^2y - z, xz^2 - y^2, zy - xy^2z\}$. Now only the last binomial is mismarked and using flip$(zy-xy^2z, \mathcal{G})$ the result is the reduced Gröbner basis with respect to the lexicographic order $\mathcal{G} = \{xy^2z - xy,\ xz^2 - y^2,\ xy - z\}$. Note that every monomial can not be divided by the leading terms, so all Gröbner bases are reduced and no auto-reduce is necessary.

$\Diamond$

The *reverse search tree* $T_{\succ}(I_A)$ with a given monomal order $>$ can be defined as follows:

Definition 3.3 (Reverse Search Tree) *[2] For two reduced Gröbner bases $\mathcal{G}_i$ and $\mathcal{G}_j$, $[\mathcal{G}_i, \mathcal{G}_j]$ directed from $\mathcal{G}_i$ to $\mathcal{G}_j$ is an edge of $T_{\succ}(I_A)$ if $\mathcal{G}_i$ is obtained from $\mathcal{G}_j$ by algorithm 5. The monomial x^α is lexicographically maximal among all facet binomials of $\mathcal{G}_i$, that are mismarked with respect to $>$.*

It can be shown that the reverse search tree is an acyclic graph with a unique sink and from any reduced Gröbner basis $\mathcal{G}_c$ of a toric ideal I_A, there is a unique path to the sink $\mathcal{G}_>$ [2]. Unlike the Gröbner walk procedure, there are still no weight vectors involved, but in return every facet of the Gröbner cone must be computed, which can be computationally expensive for reduced Gröbner bases with many polynomials.

Algorithm 7 Enumerating the edge graph of the Gröbner fan via reverse search [2]

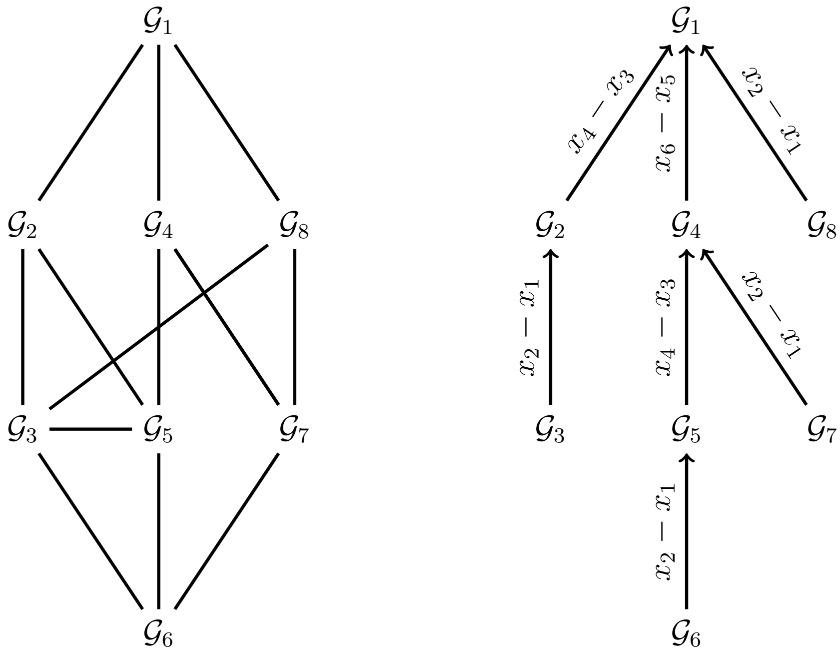
Input: Any reduced Gröbner basis $\mathcal{R}_>$ of I_A and its term order $>$
Output: All reduced Gröbner bases of I_A (all vertices of the edge graph)

$\mathcal{G} := \mathcal{R}_>$; $j := 0$; $L :=$ list of facet binomials of $\mathcal{G}$ marked by $>$
add $\mathcal{G}$ to output
repeat
    while $j < |L|$ **do**
        j := j + 1
        $\mathcal{G}' :=$ flip($\mathcal{G}$,$L[j]$)
        if $[\mathcal{G}', \mathcal{G}] \in T_>(I_A)$ **then** ▷ Check for adjacency
            $\mathcal{G} := \mathcal{G}'$; $j := 0$
            $L :=$ list of facet of $\mathcal{G}$ marked by $>$
            add $\mathcal{G}$ to output
        end if
    end while
    if $\mathcal{G} \neq \mathcal{R}_>$ **then**
        $\mathcal{G}' :=$ unique element such that $[\mathcal{G}', \mathcal{G}] \in T_>(I_A)$
        $j := 0$
        $L :=$ list of facets of $\mathcal{G}'$ marked by $>$
        repeat
            $j := j + 1$
        until the common facet of $\mathcal{G}$ and $\mathcal{G}'$ is the j-th facet of L
    end if
until $\mathcal{G} = \mathcal{R}_>$ and $j = |L|$

Example 10 Consider the reduced Gröbner basis with respect to the lexicographic order $x_1 > \ldots > x_6$

$$\mathcal{G} = \{x_1 - x_2, x_3 - x_4, x_5 - x_6, x_2^2 - 1, x_4^2 - 1, x_6^2 - 1\}.$$

Applying the breath-first search algorithm for this reduced Gröbner basis results to the following edge graph on figure 2a.
Using the reverse search method, this search tree on figure 2b results.



(a) Complete edge graph (b) Reverse search tree

Figure 2: Comparison between the breath-first search and the reverse search

The flipping binomials at figure 2b are written on the edges.

The Gröbner bases are:

$$\begin{aligned}
\mathcal{G}_1 &= \{x_1 - x_2, x_3 - x_4, x_5 - x_6, x_2^2 - 1, x_4^2 - 1, x_6^2 - 1\}\\
\mathcal{G}_2 &= \{x_1 - x_2, x_4 - x_3, x_5 - x_6, x_2^2 - 1, x_3^2 - 1, x_6^2 - 1\}\\
\mathcal{G}_3 &= \{x_2 - x_1, x_4 - x_3, x_5 - x_6, x_1^2 - 1, x_3^2 - 1, x_6^2 - 1\}\\
\mathcal{G}_4 &= \{x_1 - x_2, x_3 - x_4, x_6 - x_5, x_2^2 - 1, x_4^2 - 1, x_5^2 - 1\}\\
\mathcal{G}_5 &= \{x_1 - x_2, x_4 - x_3, x_6 - x_5, x_2^2 - 1, x_3^2 - 1, x_5^2 - 1\}\\
\mathcal{G}_6 &= \{x_2 - x_1, x_4 - x_3, x_6 - x_5, x_1^2 - 1, x_3^2 - 1, x_5^2 - 1\}\\
\mathcal{G}_7 &= \{x_2 - x_1, x_3 - x_4, x_6 - x_5, x_1^2 - 1, x_4^2 - 1, x_5^2 - 1\}\\
\mathcal{G}_8 &= \{x_2 - x_1, x_3 - x_4, x_5 - x_6, x_1^2 - 1, x_4^2 - 1, x_6^2 - 1\}.
\end{aligned}$$

A lot of edges were saved which results to less computation.

$\diamondsuit$

3.2 Degree compatible Gröbner basis

Computing the whole Gröbner fan can be very expensive and not every Gröbner basis is interesting. In this section, the degree compatible Gröbner fan is introduced and how the algorithm can be changed so that only the degree compatible Gröbner fan will be computed.

Definition 3.4 (Degree compatible Gröbner basis) *[6] A reduced Gröbner basis for an ideal I with respect to a certain monomial order is degree compatible if and only if the corresponding Gröbner cone contains the all-one vector **1**.*

Equivalent to this, the leading term of every polynomial must have the highest degree. Since a Gröbner fan is homogeneous at $\mathbb{R}^n_+$, there will be at least one degree compatible Gröbner basis. That is a special case which can be easily determined as follows.

Definition 3.5 (Only degree compatible Gröbner basis) *[6] A Gröbner basis $\mathcal{G}$ with respect to a degree compatible monomial ordering $>$ is the only degree compatible Gröbner basis for an ideal if and only if*

$$deg(x^a) > deg(x^b) \; \forall \; x^a - x^b \in \mathcal{G}.$$

This can be also described by the all-one vector that lies completely in a Gröbner cone of a Gröbner basis $\mathcal{G}$. It follows that the all-one vector does not intersect with any facets if there is only one degree compatible Gröbner basis.

The algorithms 6 and 7 can be adapted in order to compute only the degree compatible Gröbner fan.
The breadth-first search now needs a degree compatible Gröbner basis as an input. This can be achieved by applying the Buchberger Algorithm with a degree compatible monomial, for example the *grlex* order. After that it is required that the Gröbner basis is checked if its is the only degree compatible basis. Also the only facet binomials $x^a - x^b$ which are allowed to be "flipped" are the binomials that satisfy the condition $\mathrm{deg}(\mathrm{x}^a) = \mathrm{deg}(\mathrm{x}^b)$.

The reverse search tree can be deployed as in definition 3.3 but with the restriction that $\mathrm{deg}(\mathrm{x}^a) = \mathrm{deg}(\mathrm{x}^b)$ must be satisfied to traverse the degree compatible Gröbner fan. It can be ensured by [6] that at least one such facet binomial will be found. The sink of the reverse search tree contains binomials that are not mismarked with respect to some degree compatible monomial order.

3.3 Linear Codes over Prime Fields

This work is focused on computing the Gröbner fan for linear codes. Now the mathematic background of the Gröbner fans is given, the linear codes and code ideals have to be defined to give a connection between these two topics. Let $\mathbb{F}$ be a finite field and let n and $k \in \mathbb{N}$ with $n \geq k$.

Definition 3.6 (Linear Code) *[5] A linear code of length n and dimension k over $\mathbb{F}$ is the image $\mathcal{C}$ of a injective linear mapping $\phi : \mathbb{F}^k \to \mathbb{F}^n$.*

Such a code will be denoted as an $[n, k]$ code and its elements are called codewords. The codewords are written as row vectors. The Code $\mathcal{C}$ can alternatively be described as a row space matrix of $G \in \mathbb{F}^{k \times n}$. The rows of G form a basis of $\mathcal{C}$. The matrix G is also called *generator matrix* for $\mathcal{C}$.

Definition 3.7 (Standard form) *[5] A $[n, k]$ code $\mathcal{C}$ is in standard form if it has a generator matrix like $G = (I_k|M)$, where I_k is the $k \times k$ matrix.*

Example 11 Consider the binary $[7, 4]$ Hamming Code with its generator matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

The codeword c of the word x is obtained with the vector-multiplication

$$xG = c.$$

Let x be $(1,0,1,0)$, then the codeword c results to:

$$(1,0,1,0) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} = (1,0,1,0,0,0,1)$$

$\diamondsuit$

Two codes are equivalent if one generator matrix can be obtained from the other by permuting columns and rows. It follows that every linear code is equivalent to a linear code in standard form [5].

A linear Code $\mathcal{C}$ can be *punctured* by deleting individual code symbols. This reduces the length of the code and rises the data rate for a transmission of a code.

3.4 Code Ideals

In this section, the linear codes and the Gröbner bases come together.
Each linear code $\mathcal{C}$ can be associated to a binomial ideal [6]. Let $\mathcal{C}$ be a $[n,k]$ code and let $\mathbb{K}[\mathbf{x}] = \mathbb{K}[x_1,\ldots,x_n]$. Then the *code ideal* can be defined as follows:

Definition 3.8 (Code Ideal) *[6] A code ideal $I(\mathcal{C})$ is the union between the toric ideal and a nonprime ideal I_p, such that*

$$I_{\mathcal{C}} = \langle \boldsymbol{x}^c - \boldsymbol{x}^{c'} | c - c' \in \mathcal{C} \rangle + I_p,$$
$$\textit{where } I_p = \langle x_i^p - 1 | 1 \leq i \leq n \rangle.$$

Example 12 Let $\mathcal{C}_1$ be a binary $[6,3]$ code with the generator matrix

$$G_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

The associated code ideal $I(\mathcal{C})$ leads to:

$$\begin{aligned} I(\mathcal{C}) = \{&x_1 - x_5,\ x_2 - x_4x_5x_6,\ x_3 - x_5\} \cup \\ &\{x_1^2 - 1,\ x_2^2 - 1,\ x_3^2 - 1,\ x_4^2 - 1,\ x_5^2 - 1,\ x_6^2 - 1\} \end{aligned}$$

Note that the terms $x_1^2 - 1,\ x_2^2 - 1,\ x_3^2 - 1$ are divisible by the leading terms of the toric ideal. The reduced Gröbner basis $\mathcal{G}_>$ with respect to the lexicographic ordering $>$ with $x_1 > \cdots > x_6$ is:

$$\mathcal{G}_> = \{x_1 - x_5,\ x_2 - x_4x_5x_6,\ x_3 - x_5\} \cup \{x_4^2 - 1,\ x_5^2 - 1,\ x_6^2 - 1\}$$

Puncturing the fourth symbol x_4 results to new reduced Gröbner base

$$\mathcal{G}_> = \{x_1 - x_5,\ x_2 - x_5x_6,\ x_3 - x_5\} \cup \{x_5^2 - 1,\ x_6^2 - 1\}.$$

Note that the term $x_4^2 - 1$ was auto-reduced.

$\Diamond$

4 Software

This section is all about the practical part of this work. At first, an accurate description is presented of how the software can be compiled and used for own demands. Secondly, the software is tested on some randomly generated linear codes. The number of degree compatible and all Gröbner bases are presented and a comparison of the operational time against Gfan [7] is presented.
This software, called CIDGEL (Code Ideal degree compatible Gröbner bases enumerating from Linear Codes), is a re-implementation of TiGERS [2], that is why the software is written in C. All features that are needed for the code ideals were added, also the adapted algorithms for computing degree compatible Gröbner bases with reverse search and breath-first search were implemented. Additional features are explained in Section 4.2.

4.1 Data Structures

With the special attribute that code ideals only contain binomials and reduced Gröbner bases always have the coefficient 1, only the exponent vectors representing the monomials have to be stored.

Listing 1: Data structure of binomials [2]

```c
typedef struct bin_tag *binomial;
struct bin_tag{
    int *exps1;
    int *exps2;
    int *E;
    int ff;
    int bf;
    binomial next;
};
```

The pointer *exps1* stores the exponent vector of the first monomial and *exps2* does it with the second monomial. The integer *ff* is a flag that shows if a binomial is a facet binomial or not and *bf* tells if there is a monomial or binomial. The pointer *binomial next* indicates that binomials are linked together like a linear list, which is necessary to describe ideals and Gröbner bases. The pointer E points to *exps1* and *exps2* and is used for allocating and deallocating space.

The next code snippet shows the other important data structure. Again, it is a linked list like the binomials with the purpose to connect all reduced Gröbner together, which is needed for the breath-first search.

The first four integers show off the identification number of the vertex of the edge graph, the number of facet binomials, number of binomials and the highest degree. The pointer *next* is a pointer to the next generating set of the linked list. The binomial *cache_edge* and the pointer *cache_vtx* store the caching information in order not to recompute every facet binomial in the reverse search.

Listing 2: Data structure of generating sets

```
typedef struct gset_tag *gset;

/* Linked List of gset_tag which contiains the binomial and the
   caching informations*/
struct gset_tag{
    int id;
    int nfacets;
    int nelts;
    int deg;
    binomial bottom;
    binomial cache_edge;
    struct gset_tag *cache_vtx;
    struct gset_tag *next;
};
```

4.2 Manual

This software was programmed and evaluated with Linux, so at first, it is needed to compile the software. The makefile is given and it only takes the console, moving to the direction of the folder and typing 'make'.
All inputs, outputs, options and flags are passed with the command-line arguments. At first it is useful to run the program with the purpose to print the help-message only with the command : `./cidgel -h`.

Listing 3: Code Snippet of the help-message

```
static char *helpmsg[] = {
    "Function: Enumerate all or d.c Groebner bases of a code ideal I(C).",
    " \n",
    "Options:\n",
    " -h print this message\n"
    " -i (filename) set file name for input [default: stdin]\n",
    " -o (filename) set file name for output [default: stdout]\n",
    " -m (filename) set file name for code-matching \n",
    " -R only compute root of tree \n",
    " -r compute all grobner bases [done by default]\n",
    " -C turn partial caching on [done by default]\n",
    " -c turn partial caching off \n",
    " -T print edges of search tree \n",
    " -t do not print edges of search tree [assumed by default]\n",
    " -L print vertices by giving initial ideals\n",
    " and printing facet biomials.\n",
    " -l print vertices as grobner bases [done by default]\n",
    " -F Use only linear algebra when testing facets [default]\n",
    " -f use FLIPPABILITY test first when determining facets\n",
    " -e use exhaustive search instead of reverse search\n",
    " -E use reverse search [default]\n",
    " -d degree compatible Groebner bases only \n",
    " -n do not print vertices or edges \n",
    " -p calculate Groebner fans of punctured codes \n",
    NULL
};
```

The listing above shows all options that are available. These flags can be passed in an arbitrary order to the program. It is necessary to write a matrix and storing it into a file to give the program an input.

For example, the data has the name 'Example-input', has the content of listing 4 and is in the same folder as the compiled software.
Note that two inputs for a program call are possible in order to check if the two codes are equivalent. It is possible to compare the whole or the degree compatible Gröbner fan. It will be checked if the Gröbner fans have the same amount of polynomials, facets and degree for each Gröbner basis. This provides a necessary but not a sufficient condition for the Code Equivalence Problem.

Listing 4: Example-input

```
M: { 6 10 2 :
1 0 0 0 0 0 0 0 1 1
0 1 0 0 0 0 1 1 0 0
0 0 1 0 0 0 1 1 1 1
0 0 0 1 0 0 0 1 1 1
0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 1 0 1 0 1
}
```

This matrix is a generator matrix for a binary $[10, 6]$ code in the essential standard form. The first number in the first row gives the code dimension, the second tells the length of the codeword and the last number indicates in which primary field the generator matrix shall be evaluated.

For example, the degree compatible Gröbner bases of this generator matrix without printing the Gröbner bases shall be written in a output-file called `Example-output`. Additionally the linear programming shall be left out, then the program call is: `./cidgel -i Example-input -o Example-output -d -n -f`. The user do not has to specify an output file, the result will be printed in the console then.

Listing 5: Snippet of the example output

```
R: 10
G: {a−i*j, b−g*h, c−g*h*i*j, d−h*i*j, e−g, f−h*j,
         g^2−1, h^2−1, i^2−1, j^2−1}

 Enumerating degree compatible Groebner bases
   using reverse search
   taking input from Example−input
   with partial caching
   using wall ideal pretest for facet checking

Number of Groebner bases found 216
Number of edges of state polytope 792
max caching depth 10
max facet binomials 18
min facet binomials 12
max binomials in GB 41
min binomials in GB 40
max degree 3
min degree 2
Example−input: Reverse Search, Caching, A−pretest,
time used (in seconds) 42.16
```

4.3 Computational experience

In this section the difference between degree compatible Gröbner bases and all Gröbner bases from a linear code are studied. Furthermore, the computational time for all Gröbner bases is compared against Gfan [7]. The linear codes were randomly generated. All codes in this evaluation were binary.

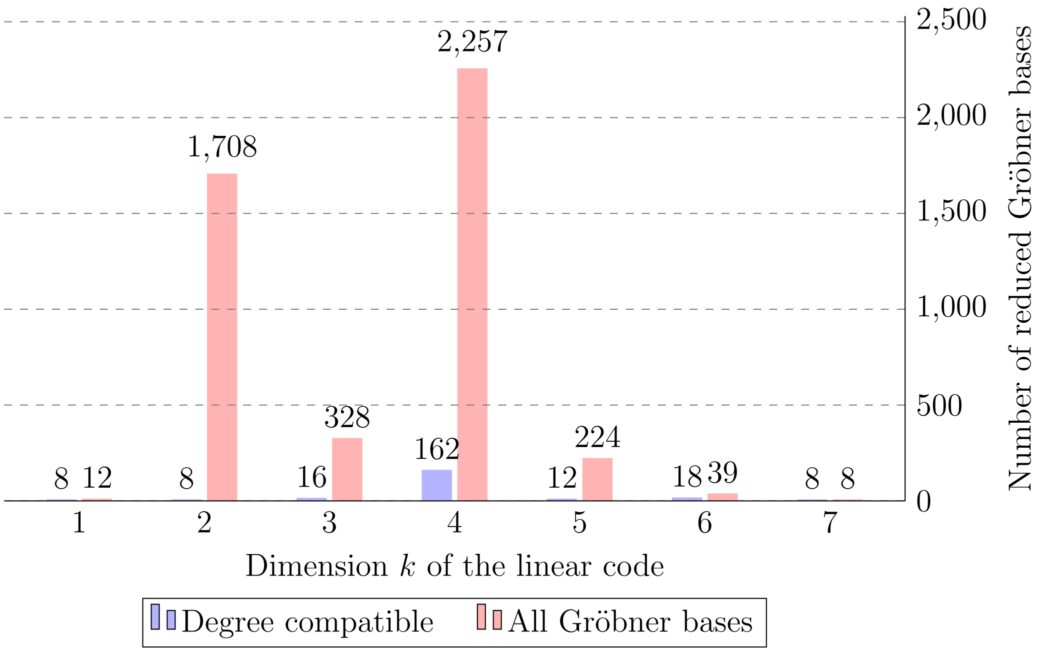


Figure 3: Comparison between the numbers of degree compatible and all Gröbner bases of binary linear codes with the length 8

Figure 3 shows a remarkable difference between all reduced Gröbner bases of a code ideal and the degree compatible Gröbner bases. The next table shows the computational time of the randomly generated codes.

Table 1: Computational time in seconds

$[n,k]$ Code	CIDGEL d.c.	CIDGEL	Gfan
$[8,1]$	0.01	0.011	0.239
$[8,2]$	0.206	10.198	38.127
$[8,3]$	0.08	0.688	6.32
$[8,4]$	7.743	25.86	47.748
$[8,5]$	0.19	0.608	3.588
$[8,6]$	0.029	0.039	0.553
$[8,7]$	0.009	0.009	0.116
$[9,4]$	9.27	727.91	982.56
$[9,5]$	4.72	18.89	59.65
$[9,6]$	0.22	0.81	4.45
$[10,6]$	87.92	277.81	380.04

Table 1 shows that the software CIDGEL can be way faster than Gfan for computing all reduced Gröbner bases from a code ideal. The reason is the special binomial structure of the code ideals and the fast algorithms that can be linked with. Gfan is a software for Gröbner bases with more general structure. Even though the amount of all Gröbner bases are much more than the degree compatible Gröbner bases, the computational time does not hold proportionally to the amount. The computational time depends mostly on the computation of facets, see page 16. The degree compatible Gröbner bases mostly have more binomials than all other Gröbner bases, that is why the computational time for a code ideal with a few Gröbner bases with high cardinality will last longer than for a code ideal with many Gröbner bases with low cardinality.

For some linear codes with a length more than 9 the computation took more than 10 hours for the complete Gröbner fan. The reason is that the amount of reduced Gröbner bases and the cardinality of each basis will be greater if the codewords will be longer. Nevertheless, the computation of the degree compatible Gröbner fan is reasonable for a certain length.

In some cases the amount of the degree compatible and all Gröbner bases can rise enormously. A randomly generated $[9,2]$ code had 8 degree compatible Gröbner bases and overall 295,863 Gröbner bases. Computing the whole fan took 24,471.32 and the degree compatible fan only took 1.08 seconds.

4.4 Documentation and electronic availability

A HTML-based documentation of the software is created with the help of doxygen [8]. The HTML-file "`index`" in the path `docu/html` is the mainpage. The software is also available under my repository
`https://github.com/smdr2670/CIDGEL`.

5 Future Work

Even if the algorithms are well suited for the given problems, this software can be improved. The algorithms themselves may already have mathematically the best performance. Section 4.3 shows that the computation of facets is the most expensive step, so the performance can be heavily improved by enhancing the linear programming. The software uses a built-in LP solver mentioned in [2], but extern LP-solvers like the GLPK (GNU Linear Programming Kit) could be more efficient.
The software package CaTS [9] enumerates all reduced Gröbner bases for a lattice Ideal. CaTS uses an external LP-Solver too to improve the performance.

Another approach will be parallelization of the computation of facets. To compute the facets, a certain amount of linear programs have to be set up, which can be solved independently. Every processor core of the computing unit can solve the linear programs.
If a LP solver is thread-safe, both ideas can be put together and lead to a large speedup.

6 Conclusion

In this thesis a software was introduced to compute Gröbner fans for linear codes. Firstly, the mathematical background and concepts were discussed with the purpose to develop the software, with the hope that it might be useful for researches and other thesis that only needs the degree compatible Gröbner fan.

In my knowledge, no other software provides this useful feature to compute the degree compatible Gröbner fan. Computing the degree compatible can be dramatically faster than computing the whole Gröbner fan.

The given data structures and the well written algorithms from TiGERS [2] made it easy to extend the software with a lot of new features in terms of linear codes and the degree compatible Gröbner fan. The computational experience in section 4.3 shows that CIDGEL can be way faster than Gfan for computing the whole Gröbner fan. Computing the degree compatible Gröbner fan with CIDGEL leads to an additional speedup.

Bibliography

[1] Karl-Heinz Zimmermann, *Algebraic Statistics.* Books on Demand GmbH, Norderstedt, July 2009.

[2] Birkett Huber and Rekha R. Thomas, "Computing Gröbner Fans of Toric Ideals," Institute for Defense Analysis, United States, Department of Mathematic, University of Washington, Tech. Rep., 1999.

[3] Mehwish Saleemi, "Coding Theory via Groebner Bases," Ph.D. dissertation, Technische Universität Hamburg-Harburg, 2012.

[4] David R. Cox, John Little and Donal O'Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[5] Natalia Dück and Karl-Heinz Zimmermann, "Universal Gröbner bases for binary linear codes," *International Journal of Pure and Applied Mathematics*, 2013.

[6] Natalia Dück, "Computation of the d.c. Gröbner fan," April 2014, Tasksetting for the bachelor thesis.

[7] Anders Nedergaard Jensen. (2005, April) The gfan homepage. [Online]. Available: http://home.imf.au.dk/jensen/software/gfan/gfan.html

[8] Dimitri van Heesch. (2014, April) Doxygen. [Online]. Available: http://www.stack.nl/~dimitri/doxygen/

[9] Anders Nedergaard Jensen. (2003, November) Cats version 2.2: A user's manual. [Online]. Available: http://www.soopadoopa.dk/anders/cats/cats.html