Contents

1 Introduction **2**

2 Mathematical Background **3**
2.1 Monomials . . . 3
2.2 Monomial Order . . . 3
2.3 Ideals . . . 5
2.4 Division Algorithm . . . 6
2.5 Groebner basis . . . 8
2.5.1 Definition of a Groebner basis . . . 8
2.5.2 Computation of a Groebner basis . . . 8
2.6 Groebner fans . . . 11
2.7 Toric Ideals . . . 13
2.8 Enumerating Groebner fans . . . 13
2.8.1 Breadth first search . . . 16
2.8.2 Reverse Search tree . . . 16
2.9 Degree compatible Groebner . . . 19
2.10 Linear Codes over Prime Fields . . . 21
2.11 Code Ideals . . . 23

3 Software **25**
3.1 Data Structures . . . 25
3.2 Manual . . . 27
3.3 Computational experience . . . 29
3.4 Documentation and electronic availability . . . 29

4 Future Work **30**

5 Conclusion **31**

1 Introduction

2 Mathematical Background

In this chapter a mathematical basis is systematically approached to give the reader an understanding to Groebner Bases and obtaining by the Flipping-Algorithm which is needed later.
In the first section monomials are revisited. The second section explains how monomials can be mathematically ordered. After that Ideals are defined over polynomial rings and a summary on Groebner bases and Groebner fans for ideals is presented.

2.1 Monomials

First of all, the basic components of a polynomial ring has to be explained. This forms the basis of

Definition 2.1 (Monomial)**.** *A* monomial m *is a product of variables over a finite field* $\mathbb{K}$*, denoted by* $\mathbb{K}[X_1, X_2, \cdots X_n]$ *of the form* $X_1^{u_1} X_2^{u_2} \cdots X_n^{u_n}$*, 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)**.** *A polynomial f is a finite linear combination with coefficients* $c_u \in \mathbb{K}$ *multiplied with monomials.*

$$f = \sum_{u} c_u X^u$$

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

2.2 Monomial Order

It is necessary to arrange the terms of a polynomial in order to compare every pair of polynomials. That is important for dividing polynomials in

the finite field $\mathbb{K}[X_1, X_2, \cdots X_n]$

Definition 2.3 (Term Ordering). *A monomial order is a relation $>$ on the set of all monomials in $\mathbb{K}[x]$ such that* [2] *holds. 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 $mm_1 > mm_2$ for any monomial m*

Two commonly used term orders are the following. Let u and v be elements of $\mathbb{N}_0^n$, such that [2]

Lexicographic Order $u >_{lex} v$ 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 $u >_{grlex} v$ if $deg(u) > deg(v)$ or if $deg(u) = deg(v)$ and $u >_{lex} v$

Example Let $m_1 = 4x^2y^4z^3$ 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, $m_1 >_{lex} m_2$ and $deg(m_1) > deg(m_2)$ so that $m_1 >_{grlex} m_2$

Weight vectors

In order to compare monomials with a generic vector $(a_1, \ldots, a_n) \in \mathbb{R}_{\geq 0}^n$, the dot product has to be build. If a tie occurs, some other fixed monomial order has to be used. Note that the standard monomial orders can be expressed as weight vector. The lexicographic order needs for instance all canonical unit vectors.

Leading term

Given a term order $>$, each non-zero polynomial $f \in \mathbb{K}[x]$ has a unique leading term, denoted by $lt(f)$, given by the largest involved term with respect to the term order.
If $lt(f) = cX^u$, where $c \in \mathbb{K}$, then c is the leading coefficient of f and X^u is the leading monomial(lm).[2]

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

2.3 Ideals

Definition 2.4 (Ideal)**.** *An ideal I is collection of polynomials* $f_1, \cdots, f_s \in \mathbb{K}[X_1, \cdots, X_n]$ *and polynomials which can be built from these with multiplication with arbitrary polynomials and linear combination, such as* [1]*:*
This is called an Ideal generated by $f_1, \cdots, f_s$

It satisfies:

- $$\langle f_1, \cdots, f_s \rangle = \left\{ \sum_{i=1}^{s} h_i f_i \mid h_1, \cdots, h_s \in \mathbb{K}[X_1, \cdots, X_n] \right\}$$
- $0 \in I$
- 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 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 $f = y \cdot f_1 + x \cdot f_2, f \in I$

⧫

Definition 2.5 (Binomial Ideal). *A binomial ideal $I \in \mathbb{K}[X_1, \cdots, X_n]$ is a polynomial Ideal, generated by binomials. A binomial is a linear combination of two monomials.*

2.4 Division Algorithm

The reader already may determine if a polynomial p lies in an Ideal I in polynomial ring with one variable. This can be achieved with the help of the polynomial division. If result has no remainder, p lies in I But in a ring with several variables like $\mathbb{K}[X_1, X_2, \cdots X_n]$ the usual division algorithm can not work. A generalized algorithm is needed. The main goal now is to divide $g \in \mathbb{K}[X_1, \cdots, X_n]$ by $f_1, \ldots, f_s \in \mathbb{K}[X_1, \cdots, X_n]$, 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 $r \in \mathbb{K}[X_1, \cdots, X_n]$ This is possible with the Theorem mentioned at [3]

Theorem 2.1 (Division Algorithm in). *Fix a monomial > on $\mathbb{Z}^n_{\geq 0}$ and let $F = (f_1, \ldots, f_s)$ be an ordered s-tuple of polynomials in $\mathbb{K}[X_1, \cdots, X_n]$. Then every $f \in \mathbb{K}[X_1, \cdots, X_n]$ can be written as*

$$f = a_1 f_1 + \ldots + a_s f_s + r$$

where $a_i, r \in \mathbb{K}[X_1, \cdots, X_n]$, and either $r = 0$ or r is a linear combination, with the coefficients in $\mathbb{K}$, none of which is divisible by any of $\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_s)$. The remainder of f on division by F is r. Furthermore, if $a_i f_i \neq 0$, then $deg(f) \geq deg(a_i f_i)$

Example

Example

Algorithm 1 Division Algorithm

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

$h_1 \leftarrow 0, \cdots, h_m \leftarrow 0$
$r \leftarrow 0$
$s \leftarrow f$
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**

$$s \leftarrow s - \frac{LT(s)}{LT(f[i])} * f_i$$

            $h_i \leftarrow h_i + LT(s) / LT(f_i)$
            division_occured = false
        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

The last example shows that is still possible to obtain a nonzero 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 Groebner basis

To solve the idea membership problem a "good" generating set for an Ideal I is needed. It would be helpful when the remainder r on division is uniqueley determined and the condition $r =$ is equivalent to the membership in the ideal. So the definition from $[KHZ]$ might be useful.

2.5.1 Definition of a Groebner basis

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

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

2.5.2 Computation of a Groebner basis

In order to obtain a Groebner basis of an arbitrary basis $f_1, \ldots, f_n$ with an arbitrary monomial order $\geq$ of an Ideal I, an algorithm is needed. This algorithm is called Buchberger-Algorithm. The main 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-Pair of (f_i, f_j) vanishes.
Let the polynomials $f, g \in \mathbb{K}[X_1, \cdots, X_n]$ 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)**.** $[KHZ]$ *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 Consider the polynomials the polynomial ring $\mathbb{K}[x,y,z]$ with the basis $\{f,g\} = \left\{xy^2 - xz + y, xy - z^2\right\}$ with respect to the lexicographic order.
Forming the S-Polynomial leads to:

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

$$= \frac{xy^2}{xy^2} \cdot \left(xy^2 - xz + y\right) - \frac{xy^2}{xy} \cdot (xy - z)$$

$$= -xz - yz + y$$

♦

The S-Polynomial is not zero and is not disvisible by the leading terms of f or g. That means the Basis given in the example is not a Groebner basis. This can be deduced by the Buchbergers criterium.

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

Now that the meaning of the S-Polynomial is clear the Buchberger algorithm can be defined.

Algorithm 2 Buchbergers Algorithm

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

```
1:  G ← F
2:  repeat
3:      G' ← G
4:      for each pair f_i and f_j in G , i ≠ j do
5:          S ← S(f_i, f_j)^{G'}          ▷ S-Polynomial with the basis of G'
6:          if G ≠ 0 then
7:              G ← G ∪ {S}
8:          end if
9:      end for
10: until G = G'
```

This algorithm is correct and terminates.[KHZ]

However, a Groebner basis is not unique. A arbitrary polynomial can be added to a Groebner basis and it is still a Groebner basis. Fortunalety a each nonzero Ideal in $\mathbb{K}[X_1, \cdots, X_n]$ has a unique *reduced* Groebner basis.

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

2.6 Groebner fans

Groebner 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 Groebner basis, the length or the degree of the elements. So it will be helpful if all possible Groebner basis of a fixed ideal can be collected together.
[$Cox, O'Shea$] shows that the collection is finite.

Definition 2.10 (Groebner fan)**.** [$Cox, O'Shea$] *A Groebner fan of an Ideal I constist of finitley many closed convex polyedral 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 Groebner fan to a given Ideal, consider the marked Groebner basis $G = \{g_1, \cdots, g_t\}$ of the Ideal I. A marked Groebner basis is a Groebner basis where each $g \in G$ has an identified leading term, such that G is a monic Groebner basis with respect to some monomial $>$ order selecting those terms. More informally, where all leading terms in G are marked.
The elements of G g_i can be written as

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

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

So the cone of a Groebner basis can be written as [*CoxOshea*]

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

example Consider the Ideal from [*CoxO'Shea*] with $I = \langle x^2 - y, xz - y^2 + yz \rangle \in \mathbb{Q}[x, y, z]$. Note that the ring is 3-dimensional so that Groebner fan can be plotted in the positive orthant $\mathbb{R}^3_+$.
The marked Groebner basis with respect to the *grevlex* order with $y > z > x$ is

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

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

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

This is the first cone of the Groebner fan and can be drawn in the positive orthant sliced of the plane of $a + b + c = 1$ for visuality.

⧫

This figure shows that the Groebner fan is not complete, since the cone does not cover the whole positive orthant. In this example, the other reduced Groebner basis can be obtained by applying the Buchberger Algorithm with common term orders to the Ideal I. If the computed cones still are not the the whole positive orthant, then a further computation with weight vectors are necessary.

This is strategy is reasonable for a small example like above. In genearal, the whole Groebner fan can be computed with the Groebner walk. See [*CoxOShea*] for further details.
An inexpensive way to obtain all reduced Groebner bases of a special ideal, the Code Ideal, which will be explained later.

2.7 Toric Ideals

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

Definition 2.11 (Toric Ideal)**.** [*Dueck Journal*] *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

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

2.8 Enumerating Groebner fans

In this section, 2 algorithms will be explained in order to enumerate the Groebner fan. For the purpose to compute all Groebner bases from a toric

ideal I_A , it is necessary to search the *edge graph* of a Groebner fan.
Two reduced Groebner bases with respect to a term order covered by the generic weight vectors c_1,c_2 are said to be adjacent if the two Groebner cones share a common *facet*.

Definition 2.12 (Facet Binomial)**.** *[TiGERS] The binomial $x^{\alpha_k} - x^{\beta_k} \in \mathcal{G}_c$ is a facet binomial of $\mathcal{G}_c$ if and only if there exists a $u \in \mathbb{R}^n$ which satisfies :*

$$\{\alpha_i \cdot u > \beta_i \cdot u : i = 1, \cdots, t, i \neq k\}$$

$$\{\beta_k \cdot u > \alpha_k \cdot u\}$$

In the example (with the Gröebner fan) the facet binomials of a Gröbner cone determine the border to an other Gröber cone. 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.

Algorithm 3 Local change of reduced Gröbner bases in I_A $[TiGERS]$

Require: Reduced Gröbner basis $\mathcal{G} = \{\underline{x}_k^a\}$ of I_A
A prescribed facet binomial $\underline{x}_i^a - x_i^b \in \mathcal{G}$ ▷ The weight vector inducing $\mathcal{G}$ is generic and the leading terms are underlined

Ensure: 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 := Computed reduced Gröbner basis with respect to the new marking ▷ This can be done with Buchberger Algorithm.
4: $\mathcal{G}' = \{\underline{x}_i^b - x_i^a\}$
5: **for each** monmial 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. $[Tigers]$ 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.

example

2.8.1 Breadth first search

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

Algorithm 4 Enumerating the edge graph of the Gröbner fan via breath-first search $[TiGERS]$

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

```
1:  Todo := [G_0]
2:  Verts := []
3:  while Todo ≠ ∅ do
4:      G := first element in (Todo)
5:      Remove G from Todo
6:      add G to Verts
7:      determine list L of facet binomials of G   ▷ With linear programming
8:      for each x^α − x^β ∈ L do
9:          G' = flip(G, x^α − x^β)
10:         if G' ∉ Todo ∪ Verts then
11:             add G' to Todo
12:         end if
13:     end for
14: end while
15: 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.

2.8.2 Reverse Search tree

In this section, a memoryless algorithm is presented which has the benefit that it runs linear depending on the size of output. The main idea is to enumerate the edge graph with depth first reverse search. The result will be

a directed subgraph of the edge graph, called reverse search tree $T_{\succ}(I_A)$. But before that, the meaning of a mismarked polynomial must be clear.

Definition 2.13 (Mismarked Polynomial)**.** [*Tigers*] *A polynomial f that has been marked with respect to the monomial order $\succ$ is mismarked with respect to the monomial order $\succ$*

Example Consider the reduced Gröbner base $\mathcal{G} = \{x^2y - z, y^2 - xz, zy - xy^2z\}$ with a certain monomial order $\succ$, which is not the lexicograpic order. Then, the second and last term are clearly mismarked with respect to $\succ_{lex}$.
Applying the flip-procedure $(\mathcal{G}, y^2 - xz)$ leads to the Groebner base $\mathcal{G} = \{x^2y - z, xz^2 - y^2, zy - xy^2z\}$. Now only the last binomial is mismarked and using flip($\mathcal{G}, zy - xy^2z$) the result is the reduced Groebner 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 Groebner bases are reduced and no auto-reduce is necessary.

♦

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

Definition 2.14 (Reverse Search Tree)**.** *TiGERS For two reduced Groebner 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 $\mathcal{G}_{\rangle}$ is obtained from $\mathcal{G}_{|}$ by the procedure flip($\mathcal{G}_i$,$x^\alpha - x^\beta$). x^α is lexicographically maximal among all facet binomials of $\mathcal{G}_i$, that are mismarked with respect to $\succ$.*

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

Algorithm 5 Enumerating the edge graph of the Gröbner fan via reverse search [*TiGERS*]

Require: Any reduced Gröbner basis $\mathcal{R}_\succ$ of I_A and its term order $\succ$
Ensure: All reduced Gröbner bases of I_A, (all vertices of the edge graph)

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

Example Consider the Ideal with the reduced Groebner basis with respect to the lexicographic order $x_1 \succ \ldots \succ x_6$

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

Applying the breath-first search algorithm for this reduced Groebner basis results to the following edge graph.

Using the reverse search method, this search tree results.

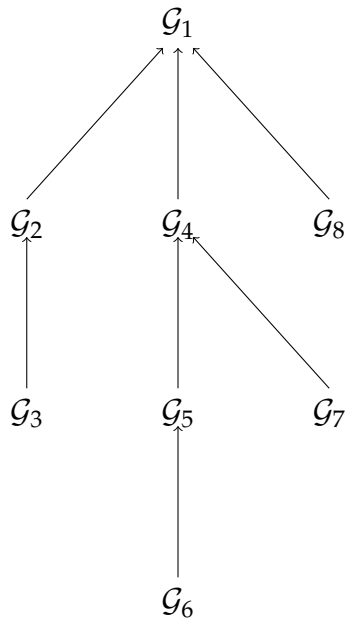

Figure 1: The reverse search tree for $\mathcal{G}_1$

Even for this small example a lot of edges were saved. 2.

2.9 Degree compatible Groebner

Computing the whole Groebner fan can be very expensive and not every cone is interesting. In this section, the degree compatible Groebner fan is

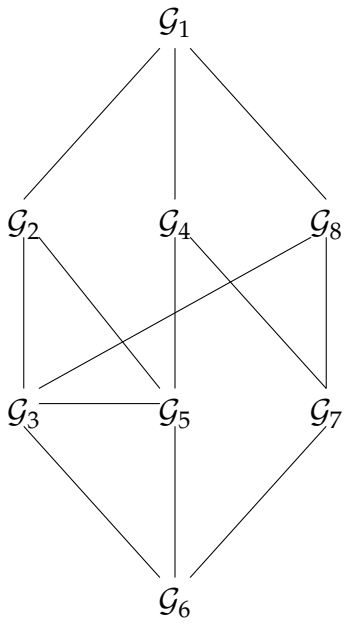

Figure 2: The complete edge graph via breadth first search

introduced and how the algorithm can be changed so that only the degree compatible Groebner fan will be computed.

Definition 2.15 (Degree compatible Groebner basis)**.** [*dueckpaper*] *A reduced Groebner basis for an ideal I with respect to a certain monomial order is degree compatible if and only if the corresponding Groebner cone contains the all-one vector 1.*

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

Definition 2.16 (Only degree compatible Groebner basis)**.** [*dueckpaper*] *A Groebner basis $\mathcal{G}$ with respect to a degree compatible monomial ordering $\succ$ is the*

only degree compatible Groebner 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 Groebner cone of a Groebner basis $\mathcal{G}$. It follows that the all-one vector does not cut any facets if there is only one degree compatible Groebner basis.

The algorithms 4 and 5 can be adapted in order to compute only the degree compatible Groebner fan.

The breadth-first search now needs a degree compatible Groebner 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 Groebner basis is checked if its is the only degree compatible basis. Also the only facet binomials x^a - x^b which are allow to be "flipped" are the binomials that fulfills the condition deg(x^a) = deg(x^b).

The reverse search tree can be deployed as in Definition 2.14 but with the restriction that deg(x^a) = deg(x^b) must be fulfilled to traverse the degree compatible Groebner fan. [*dueckpaper*, *Lemma*2.2] guarantees 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 the grlex order.

2.10 Linear Codes over Prime Fields

This work is concentrated on computing the Groebner fan of a linear code. Now the mathematic background of the Groebner 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 2.17 (Linear Code)**.** *[DueckJournal] 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 als 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 row space matrix of $G \in \mathbb{F}^{k \times n}$. The rows of G form a basis of $\mathcal{C}$. G is also calles *generator matrix* for $\mathcal{C}$.

Definition 2.18 (Standard form)**.** *[dueckjournal] 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 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 code 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:

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

♦

Two codes are equivalent if one generator matrix can be obtained from the other by permutating columns and rows. It follows that every linear code is

equivalent to a linear code in standard form.[*dueckjournal*]

A linear Code $\mathcal{C}$ can be *punctured* by deleting the same coordinate i in each codeword.

2.11 Code Ideals

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

Definition 2.19 (Code Ideal)**.** [*dueckpaper*] *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 Let $\mathcal{C}_1$ be a binary $[6,3]$ code with 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:

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

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 Groebner basis $\mathcal{G}_{\succ}$ with respect to the lexicographic ordering $\succ$ with $x_1 \succ \cdots \succ x_6$ is:

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

⧫

3 Software

This section is all about the practical part of this work. At first, a brief reason why the software is implemented in C is given. Secondly, an accurate description is presented of how the software can be compiled and used for own demand. After that, the software is tested on some randomly generated linear Codes. The number of degree compatible and all Groebner bases are presented and comparision of the operational time agains Gfan[*Gfan*] is presented.

This software is a re-implementation of TiGERS [*Tigers*]. All features that are needed for the code ideals were added, also the adapted algorithms for computing degree compatible Groebner bases with reverse search and breath-first search were implemented. Additional features are explained later in Section 3.2. The programming language of choice is C because it is fast and the capability to make own data structures are simple and sufficient enough.

3.1 Data Structures

With the special attribute that code ideals only contains binomials and monomials can be represented as an exponent vector, it is useful to store this vector in to dynamic array.

Listing 1:]Data structure of binomials[Tigers]

```
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 which shows if 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 Groebner bases.

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 Groebner 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. Next is a pointer to the next generating set of the linked list.

Listing 2:]Data structure of generating sets [Tigers]

```
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;
};
```

3.2 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 where folder is and typing 'make'.
All inputs, outputs, options and flags are passed with the commandline 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 arbitrary order to the program. An example will be shown next. But before it is necessary to write a matrix and storing it into a file.

For example, the data has the name '`matrix`', has the following content and is in the same folder as the compiled software.

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 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.

Now if the degree compatible Groebner bases of this generator matrix without printing the Groebner basis shall be written in a outputfile called `Example-output`. Additionally the linear programming shall be leaved 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: Snipped of the example outsput

```
% starting GB:
F: 2
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 randomgenerator/10_6
    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
randomgenerator/10_6: Reverse Search, Caching, A−pretest,
time used (in seconds) 42.16
```

3.3 Computational experience

In this section the difference between degree compatible Groebner bases and all Groebner bases from a linear Code are researched. Furthermore, the time for all Grooebner bases is compared against Gfan.

3.4 Documentation and electronic availability

A HTML-based documentation of the software is created with the help of doxygen.[*doxygen*]

4 Future Work

5 Conclusion