rijksuniversiteit groningen

faculteit Wiskunde en Natuurwetenschappen

Gröbner bases and Graver bases used in integer programming

Masterthesis Mathematics

July 2013

Student: M. Hoekstra

Supervisors: Prof. dr. J. Top and Dr. C. Dobre

Abstract

This thesis combines topics from the field of Algebra and the field of Optimization. It will be discussed how two different algebraic concepts can be used to solve integer linear minimization problems. The first concept studied is the (reduced) Gröbner (from now on denoted as Groebner) basis of a monomial ideal in the polynomial ring over a field k, introduced in 1939 and named after W. Gröbner. We will see how we can transform a linear system $Ax = b$ to a system of polynomial equations, so that Groebner bases can be used to solve it. The second topic is the Graver basis, introduced by Jack E. Graver in 1975. An integer linear minimization programming problem where all variables are bounded from below and above is NP-hard and hence presumably cannot be solved in polynomial time. However, given the Graver basis $Gr(A)$ of the toric ideal defined by the matrix A, it can be solved in polynomial time. Moreover, $Gr(A)$ can be used to solve bounded separable convex integer minimization problems.
We implemented algorithms by Adams and Loustaunau (1994) and by Onn (2010), using Groebner and Graver basis respectively, in Maple. The results will be displayed and discussed.

Contents

1 Preliminaries

In this chapter we state some definitions and results from algebra and optimization, which will be used throughout this thesis.

1.1 Basic Algebraic Properties

Definition 1.1. *([3, p.1]) A* ***monomial*** *in the collection of variables $x_1, x_2, \dots, x_n$ is a product of the form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ where all exponents are nonnegative exponents. The* ***total degree*** *of this monomial is the sum $|\alpha| := \sum_{i=1}^{n} \alpha_i$.*

We will use a simplifying notation for monomials letting $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{Z}^n_{\geq 0}$ be the n-tuple of nonnegative exponents and then we write

$$x^\alpha = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n}.$$

In this notation, a polynomial is a linear combination of monomials with coefficients from the field k:

Definition 1.2. *A* ***polynomial*** *f in $x_1, \dots, x_n$ with coefficients in a field k is a finite linear combination with coefficients in k of monomials. We will write a polynomial f in the form*

$$f = \sum_{\alpha \in \mathbb{Z}^n_{\geq 0}} a_\alpha x^\alpha, \ a_\alpha \in k,$$

where the sum is over a finite number of n-tuples α. The set of all such polynomials forms a (commutative) ring with the usual addition and multiplication, denoted as $k[x_1, \dots, x_n]$.

When dealing with polynomials, we will use the following terminology:

Definition 1.3. *([3, p.2]) Let $f = \sum_\alpha a_\alpha x^\alpha$ be a polynomial in $k[x_1, \dots, x_n]$.*

i. We call a_α the ***coefficient*** *of the monomial x^α.*

ii. If $a_\alpha \neq 0$, then we call $a_\alpha x^\alpha$ a ***term*** *of f.*

iii. The ***total degree*** *of f, denoted by $\deg(f)$, is the maximum $|\alpha|$ such that the coefficient $a_\alpha \neq 0$.*

As an example, the polynomial $f = x^3y^2z^3 - 3x^5y^2z + 2xz - y$ has four terms and total degree $\deg(f) = 8$. In this case there are two terms of total degree. We will see in Chapter 2 how to order the monomials of a polynomial.
We say that a polynomial f divides a polynomial g if there is some $h \in k[x_1, \dots, x_n]$ such that $g = fh$.

Definition 1.4. *([3, p.3]) Given a field k and a positive integer n, we define the n-dimensional* ***affine space*** *over k to be the set*

$$k^n = \{(a_1, \dots, a_n) : a_1, \dots, a_n \in k\}.$$

Using this affine space, we can regard a polynomial as a function. A polynomial $f = \sum_\alpha a_\alpha x^\alpha \in k[x_1, \ldots, x_n]$ gives then a function

$$f : k^n \to k.$$

This means that, for a given $(a_1, \ldots, a_n) \in k^n$, we replace every x_i by a_i in the expression for f. Since all coefficients a_α lie in the field k, this new expression lies in k too.
More precisely, consider $\mathfrak{F} := \{\text{all functions } f : k^n \to k\}$. This is a commutative ring using pointwise addition/multiplication. The map $k[x_1, ..., x_n] \to \mathfrak{F}$ given by $f \mapsto [a \mapsto f(a)]$ is a ring homomorphism. In general, it is neither injective nor surjective. Is is easy to see that for example different polynomials f can map to the same $[a \mapsto f(a)]$.

Using the notions introduced before, we can make a step from algebra towards algebraic geometry, by defining the following geometric object.

Definition 1.5. *([3, p.5]) Let k be a field and let $f_1, \ldots, f_s$ be the polynomials in $k[x_1, \ldots, x_n]$. Then we set*

$$V(f_1, \ldots, f_s) = \{(a_1, \ldots, a_n) \in k^n : f_i(a_1, \ldots, a_n) = 0 \textit{ for all } 1 \leq i \leq s\}.$$

*We call $V(f_1, \ldots, f_s)$ the **affine variety** defined by the polynomials $f_1, \ldots, f_s$.*

Hence, an affine variety $V(f_1, \ldots, f_s) \in k^n$ is exactly the set of all solutions to the system of equations $f_i(a_1, \ldots, a_n) = 0$ for $i = 1, \ldots, s$. Familiar examples of affine varieties are circles, ellipses, parabolas and hyperbolas. But also the graph of a polynomial function $y = f(x)$ can be displayed as the variety $V(y - f(x))$.

Definition 1.6. *([3, p.29]) A subset $I \subset k[x_1, ..., x_n]$ is an **ideal** of polynomials if it satisfies:*

i. $0 \in I$;

ii. If $f, g \in I$, then $f + g \in I$;

iii. If $f \in I$ and $h \in k[x_1, ..., x_n]$, then $hf \in I$.

Definition 1.7. *Let $f_1, \ldots, f_s$ be polynomials in $k[x_1, ..., x_n]$. Then we set*

$$\langle f_1, \ldots, f_s \rangle = \left\{ \sum_{i=1}^{s} h_i f_i : h_1, \ldots, h_s \in k[x_1, ..., x_n] \right\}.$$

Proposition 1.8. *([3, p.29]) If $f_1, \ldots, f_s \in k[x_1, ..., x_n]$, then $\langle f_1, \ldots, f_s \rangle$ is an ideal of $k[x_1, ..., x_n]$. We call $\langle f_1, \ldots, f_s \rangle$ the **ideal generated by** $f_1, \ldots, f_s$.*

Proof. i. Let $I := \langle f_1, \ldots, f_s \rangle$. Then $0 \in I$ since $0 = \sum_{i=1}^{s} 0 \cdot f_i$ and $0 \in k[x_1, ..., x_n]$.

ii. Suppose $f, g \in I$, then $f = \sum_{i=1}^{s} p_i f_i$, $g = \sum_{i=1}^{s} q_i f_i$ with $p_i, q_i \in k[x_1, ..., x_n]$. Further $f + g = \sum_{i=1}^{s} (p_i + q_i) f_i$. Since $k[x_1, ..., x_n]$ is a ring it is closed under addition, thus $p_i + q_i \in k[x_1, ..., x_n]$. It follows that $f + g \in I$.

iii. Let $f = \sum_{i=1}^{s} p_i f_i \in I$ and $h \in k[x_1, ..., x_n]$. Then $hf = h\sum_{i=1}^{s} p_i f_i = \sum_{i=1}^{s} (hp_i) f_i$. Since $k[x_1, ..., x_n]$ is a ring it is closed under multiplication, thus we know $hp_i \in k[x_1, ..., x_n]$. Therefore $hf \in I$ and we conclude that $I = \langle f_1, \ldots, f_s \rangle$ is an ideal.

□

We will now make a connection between the concepts of a variety and an ideal. Suppose we have an affine variety $V = V(f_1, \ldots, f_s)$ introduced by Definition 1.5, then we know that the polynomials $f_1, \ldots, f_s$ vanish on V. But there might be more polynomials vanishing on V. For example, the polynomial which is a linear combination of at least two polynomials in $\{f_1, \ldots, f_s\}$. This intuitively leads us to the idea that the set of polynomials vanishing on V is an ideal.

Definition 1.9. *Let $V \subset k^n$ be an affine variety. Then we set*

$$I(V) = \{f \in k[x_1, ..., x_n] \mid f(a_1, \ldots, a_n) = 0 \text{ for all } (a_1, .., a_n) \in V\}.$$

Proposition 1.10. *([3, p.32]) If $V \subset k^n$ is an affine variety, then $I(V) \subset k[x_1, ..., x_n]$ is an ideal. We call $I(V)$ the* ***ideal of*** *V.*

Proof. i. $0 \in I(V)$ since the zero polynomial vanishes on any n-tuple from k^n, and in particular on V.

ii. If $f, g \in I(V)$ and $(a_1, .., a_n) \in V$ then we know

$$f(a_1, \ldots, a_n) = g(a_1, \ldots, a_n) = 0.$$

Therefore $f(a_1, \ldots, a_n) + g(a_1, \ldots, a_n) = 0$, and thus $f + g \in I(V)$.

iii. Let $f \in I(V)$, $h \in k[x_1, ..., x_n]$ and $(a_1, \ldots, a_n) \in V$. Then $(hf)(a_1, \ldots, a_n) = h(a_1, \ldots, a_n)f(a_1, \ldots, a_n) = h(a_1, \ldots, a_n) \cdot 0 = 0$. We conclude that $I(V)$ is an ideal.

□

One final lemma which will turn out to be useful later.

Lemma 1.11. *([5, p.80]) Let $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ be elements of a commutative ring R. Then the element $a_1 a_2 \cdots a_n - b_1 b_2 \cdots b_n$ is in the ideal $\langle a_1 - b_1, a_2 - b_2, \ldots, a_n - b_n \rangle$.*

Proof. Although there is a proof on [5, p.80], there is much shorter way to proof this lemma. It namely suffices to show that $\overline{a_1 a_2 \cdots a_n - b_1 b_2 \cdots b_n} = \overline{0}$ in the ring $R/\langle a_1 - b_1, \ldots, a_n - b_n \rangle$. Since $\overline{a_i} = \overline{b_i}$ in this quotient ring, we are done. □

1.2 Introduction to Optimization

Definition 1.12. *([1, p.15]) A* ***mathematical minimization programming problem****, or* ***optimization problem*** *has the form:*

$$\begin{aligned} &\textit{minimize} && f(x) \\ &\textit{subject to} && g_i(x) \le b_i, \quad i = 1, \ldots, m. \end{aligned} \tag{1}$$

Here the vector $x = (x_1, \dots, x_n) \in \mathbb{R}^n$ is the **optimization variable** of the problem, the function $f : \mathbb{R}^n \to \mathbb{R}$ is called **objective function**, the functions $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \dots, m$, are the (inequality) **constraint functions**, and $b_1, \dots, b_m$ are constants. The set $\{x \in \mathbb{R}^n \mid g_i(x) \leq b_i,\ i = 1, \dots, m\}$ is called the **feasible set**, often denoted as $\mathcal{F}$. A vector $x^* \in \mathcal{F}$ is called **optimal** if it provides with the smaller objective value among all vectors in $\mathcal{F}$.
We speak of a **linear programming problem** if the objective and constraint functions $g_0, \dots, g_m$ are linear, i.e. they satisfy

$$g_i(\alpha x + \beta y) = \alpha g_i(x) + \beta g_i(y)$$

for all $x, y \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$. If this is not the case, we call it a **nonlinear programming problem**.
Another class of optimization problems are the **convex programming problems**. These are problems where the objective and constraint functions are convex.

Definition 1.13. *A set C is a **convex set** if the line segment between any two points in C lies in C, i.e. if for any $x, y \in C$ and θ with $0 \leq \theta \leq 1$, we have*

$$\theta x + (1 - \theta)y \in C.$$

Definition 1.14. *([1, p.24]) The **convex hull** of a set C, denoted as convC, is the set of all convex combinations of points in C:*

$$convC = \{\theta_1 x_1 + \ldots + \theta_k x_k \mid x_i \in C; \theta_i \geq 0; i = 1, \dots, k; \theta_1 + \ldots + \theta_k = 1\}.$$

Definition 1.15. *Let $D \subset \mathbb{R}^n$. A function $f : D \to \mathbb{R}$ is a **convex function** if its domain D is a convex set and for all $x, y \in D$ and θ with $0 \leq \theta \leq 1$, we have*

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y). \tag{2}$$

Note that every linear function is a convex function, but not vice versa.
Geometrically inequality (2) means that the line segment from $(x, f(x))$ to $(y, f(y))$ lies above the graph of f. A very important property of a convex function is that any local minimum is also a global minimum. This follows from the first-order condition on convex functions, where the gradient ∇f has been defined as

$$\nabla f = \frac{\partial f}{\partial x_1} e_1 + \frac{\partial f}{\partial x_2} e_2 + \ldots + \frac{\partial f}{\partial x_n} e_n$$

with e_i the standard orthogonal unit vectors.

Proposition 1.16. *([1, p.69]) Suppose $D \subset \mathbb{R}^n$ is nonempty and open and $f : D \to \mathbb{R}$ is differentiable, i.e. its gradient ∇f exists at each point in D. Then f is convex if and only if the set D is convex and*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

holds for all $x, y \in D$.

We see that if $\nabla f(x) = 0$, then for all y in the domain of f it yields that $f(y) \geq f(x)$, and therefore x is a global minimizer of f. For the proof of this property we refer to [1, p.70]
A special kind of convex function that we will use is the separable convex function.

Definition 1.17. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called* ***separable convex*** *if*

$$f(x) = \sum_{j=1}^{n} f_j(x_j),$$

with each $f_j : \mathbb{R} \to \mathbb{R}$ convex.

1.3 Integer Programming

In this thesis, we are especially interested in integer linear minimization programming problems:

Definition 1.18. *([5, p.105])* ***Integer linear minimization programming problem*** *If $A \in \mathbb{Z}^{n\times m}$, $b \in \mathbb{Z}^n$, and $c \in \mathbb{R}^m$, we wish to find a solution $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m) \in \mathbb{N}^m$ of the system*

$$A\sigma = b,$$

which minimizes the 'cost function'

$$c(\sigma_1, \sigma_2, \ldots, \sigma_m) = \sum_{j=1}^{m} c_j \sigma_j.$$

Here, the feasible solutions (or feasible region) are the lattice points in the polyhedron

$$P = conv\{x \in \mathbb{Z}^n \text{ s.t. } Ax = b\}.$$

In the following chapter we will proceed with algebra theorems and results. Later on, we will see how the algebraic theory can help in solving integer linear programming problems as stated above (Chapter 3) and integer separable convex programming problems (Chapter 4).

2 Monomial orderings and divisibility

2.1 Monomial orderings

In the previous chapter we have introduced ideals generated by polynomials. In this chapter we will deal with the problem, given a polynomial, to determine whether this polynomial is in the ideal. This is what we will call the *Ideal Membership Problem.* To confirm that our polynomial is indeed in the ideal we need that the polynomial can be constructed out of the polynomials that generate the ideal. This is equivalent to the question whether our polynomial is divisible by elements of the basis. Divisibility turns out to be an important tool for finding elements of an ideal. Therefore we will discuss divisibility in the single-variable case and extend it to the multi-variable case. Before we can do that, we have to study how monomials are ordered. In the single-variable case it is common to order monomials by degree and we will now discuss the several possible orderings in the multi-variable case.

Definition 2.1. *([3, p.54]) A* ***monomial ordering*** *on $k[x_1, \dots, x_n]$ is any relation $>$ on $\mathbb{Z}^n_{\geq 0}$, or equivalently, any relation on the set of monomials x^α, $\alpha \in \mathbb{Z}^n_{\geq 0}$, satisfying:*

i. $>$ is a total (or linear) ordering on $\mathbb{Z}^n_{\geq 0}$;

ii. If $\alpha > \beta$ and $\gamma \in \mathbb{Z}^n_{\geq 0}$, then $\alpha + \gamma > \beta + \gamma$;

iii. $>$ is a well ordering on $\mathbb{Z}^n_{\geq 0}$. This means that every non-empty subset of $\mathbb{Z}^n_{\geq 0}$ has a least element in this ordering.

This last restriction will turn out to be very useful, since it causes that various algorithms are finite because they work with a term that decreases at each step of the algorithm. The definition tells us that there is an end in this decreasing sequence.
We will now discuss some examples of monomial orderings, starting with the Lex(icographic) Order $>_{lex}$. We assume that an element $\alpha \in \mathbb{Z}^n_{\geq 0}$ can be written as $\alpha = (\alpha_1, \dots, \alpha_n)$.

Definition 2.2. ***Lexicographic Order****. Let $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{lex} \beta$ if in the vector $\alpha - \beta \in \mathbb{Z}^n_{\geq 0}$, the left-most nonzero element is positive. We write $x^\alpha >_{lex} x^\beta$ if $\alpha >_{lex} \beta$.*

Proposition 2.3. *The lex ordering is a monomial ordering on $\mathbb{Z}^n_{\geq 0}$.*

Proof. i. This follows from the definition and the fact that the numerical order on $\mathbb{Z}^n_{\geq 0}$ is a total ordering.

ii. Suppose $\alpha >_{lex} \beta$ such that the left most nonzero element is $\alpha_k - \beta_k > 0$, and $\gamma \in \mathbb{Z}^n_{\geq 0}$. Then $(\alpha + \gamma) - (\beta + \gamma) = \alpha - \beta$, thus the left-most nonzero element is again $\alpha_k - \beta_k > 0$, therefore $\alpha + \gamma >_{lex} \beta + \gamma$.

iii. We prove this by contradiction. Suppose that $>_{lex}$ is not a well-ordering. Then there exists a nonempty subset $S \subset \mathbb{Z}^n_{\geq 0}$ that has no least element. Let $\alpha_1 \in S$, then because α_1 is not the least element of S there is an element $\alpha_2 \in S$ such that $\alpha_1 > \alpha_2$. Continuing this argument we construct a sequence, which is infinite and strictly decreasing:

$$\alpha_1 >_{lex} \alpha_2 >_{lex} \alpha_3 >_{lex} \cdots.$$

So $\alpha_l >_{lex} \alpha_{l+1}$ for all $l \geq 1$. The definition of the lex order implies that the first entries of these elements α_i form a monotonic decreasing sequence of elements in $\mathbb{Z}_{\geq 0}$. Since the numerical order on the nonnegative integers is a well-ordering, there must be an element α_k in the sequence above for which the first entries of all α_i for $i \geq k$ are equal. If we define a new sequence, identical to the previous sequence but starting at α_k, we know that the lex order will depend on the second component of elements in the sequence. By the same argument as before, we now conclude that from some α_m on, all second components of the elements α_i for $i \geq m$ are equal. Repeating this procedure, we will end up with a vector α_l with n identical entries and all vectors α_j for $j \geq l$ are equal. But this contradicts to the property of the first sequence that $\alpha_l >_{lex} \alpha_{l+1}$ for all $l \geq 1$. We conclude that $>_{lex}$ is a well-ordering.

□

Example 2.4. *Lex order with $x > y > z$ (order in the vector).*

1. *$x^2y >_{lex} xy$, since $(2,1) - (1,1) = (1,0)$ and therefore $(2,1) >_{lex} (1,1)$.*
2. *$x^2 >_{lex} xy^2z$, since $(2,0,0) - (1,2,1) = (1,-2,-1)$ and therefore $(2,0,0) >_{lex} (1,2,1)$.*
3. *Because $(1,0,\dots,0) > (0,1,0,\dots,0) > \dots > (0,\dots,0,1)$ we see that by the Lexicographic Order the variables $x_1, x_2, \dots, x_n$ are ordered in the natural way.*

For some purposes we may want to take into account the total degree of the monomials. It can be useful to order the monomials of bigger degree first. To do so, we choose the Graded Lexicographic Order $>_{grlex}$.

Definition 2.5. ***Graded Lexicographic Order.*** *Let $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{grlex} \beta$ if $|\alpha| > |\beta_i|$, or $|\alpha| = |\beta|$ and $\alpha >_{lex} \beta$.*

We see that the Graded Lexicographic order first uses the total degree to order monomials. When monomials turn out to have equal total degree, it uses the Lexicographic order to "break ties".

Example 2.6.

1. *$x^2y >_{grlex} xy$, since $|\alpha| = 2+1 > 1+1 = |\beta|$. We see that these particular monomials are ordered in the same way as by the lex order.*
2. *$xy^2z >_{grlex} x^2$, since $|\alpha| = 4 > 2 = |\beta|$. This is an example of two monomials that are ordered differently by the graded lex order than by the lex order.*

We now have seen different orderings of monomials and we will use the following terminology about polynomials under such a monomial order.

Definition 2.7. *([3, p.58]) Let $f = \sum_\alpha a_\alpha x^\alpha$ be a nonzero polynomial in $k[x_1, ..., x_n]$ and let $>$ be a monomial order.*

i. *The **multidegree** of f is*

$$multideg(f) = \max\{\alpha \in \mathbb{Z}^n_{\geq 0} : a_\alpha \neq 0\},$$

where the maximum is taken with respect to $>$.

ii. *The **leading coefficient** of f is*

$$LC(f) = a_{multideg(f)} \in k.$$

iii. *The **leading monomial** of f is*

$$LM(f) = x^{multideg(f)}.$$

iv. *The **leading term** of f is*

$$LT(f) = LC(f) \cdot LM(f).$$

For the next section we will need the following lemma.

Lemma 2.8. *([3, p.59]) Let $f, g \in k[x_1, ..., x_n]$ be nonzero polynomials. Then:*

i. *$multideg(fg) = multideg(f) + multideg(g)$.*

ii. *If $f + g \neq 0$, then $multideg(f + g) \leq \max(multideg(f), multideg(g))$. If, in addition, $multideg(f) \neq multideg(g)$, then equality occurs.*

2.2 Divisibility

Divisibility is an important tool for finding elements of an ideal. Since we are interested in ideals in $k[x_1, ..., x_n]$, we will first discuss divisibility in the single-variable case and extend it to a division algorithm in $k[x_1, ..., x_n]$.

Theorem 2.9. *[5, p.11] **Division Algorithm in** $k[x]$*
Let g be a nonzero polynomial in $k[x]$. Then for any $f \in k[x]$, there exist quotient q and remainder r in $k[x]$ such that

$$f = qg + r, \text{ with } r = 0 \text{ or } \deg(r) < \deg(g).$$

Moreover, r and q are unique.

As a result of Theorem 2.9 we have the following algorithm:

```
INPUT: $f, g \in k[x]$ with $g \neq 0$
OUTPUT: $q, r$ such that $f = qg + r$ and $r = 0$ or $\deg(r) < \deg(g)$
INITIALIZATION: $q := 0;\ r := f$
WHILE $r \neq 0$ AND $\deg(g) \leq \deg(r)$ DO
    $q := q + \frac{LT(r)}{LT(g)},\ r := r - \frac{LT(r)}{LT(g)} g.$
```

We illustrate the algorithm with the following example.

Example 2.10. *Assume $f = 3x^3 + 2x^2 - x + 1$ and $g = x^2 + x$. In the initialization we set $q := 0$ and $r := 3x^3 + 2x^2 - x + 1$. Then $r \neq 0$ and $\deg(g) = 2 \leq 3 = \deg(r)$, so we will start the while loop. In the first step we set*

$$q := q + \tfrac{LT(r)}{LT(g)} = 0 + \tfrac{3x^3}{x^2} = 3x,$$
$$r := r - \tfrac{LT(r)}{LT(g)}g = 3x^3 + 2x^2 - x + 1 - (3x)(x^2 + x) = -x^2 - x + 1.$$

Then $r \neq 0$ and $\deg(g) = 2 \leq 2 = \deg(r)$, so the algorithm will proceed with the next step. Here, we set

$$q := q + \tfrac{LT(r)}{LT(g)} = 3x + \tfrac{-x^2}{x^2} = 3x - 1,$$
$$r := r - \tfrac{LT(r)}{LT(g)}g = -x^2 - x + 1 - (-1)(x^2 + x) = 1.$$

Now $\deg(g) = 2 > 0 = \deg(r)$, so the algorithm ends and the result is $f = (3x - 1)g + 1$.

It is important to see that the monomials are ordered here by degree. So $\deg(g) > \deg(r)$ means actually that the algorithm stops when 'g is bigger than r' which means that r is not big enough anymore to be diminished by a multiple of g. When extending this algorithm to the multi-variable case we can not use this ordering of monomials but we use one of the orderings discussed in the previous subsection. Our general goal is to divide $f \in k[x_1, ..., x_n]$ by $f_1, \ldots, f_s \in k[x_1, ..., x_n]$.

Theorem 2.11. *[3, p.63]* ***Division Algorithm in*** $k[x_1, ..., x_n]$
Fix a monomial order $>$ on $\mathbb{Z}^n_{\geq 0}$ and let $F = (f_1, \ldots, f_s)$ be an ordered s-tuple of polynomials in $k[x_1, ..., x_n]$. Then every $f \in k[x_1, ..., x_n]$ can be written as

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

where $a_i, r \in k[x_1, ..., x_n]$ for all i and either $r = 0$ or r is a k-linear combination of monomials, none of which is divisible by any of $LT(f_1), \ldots, LT(f_s)$. We will call r a remainder of f on division by F. Furthermore, if $a_i f_i \neq 0$, then we have

$$multideg(f) \geq multideg(a_i f_i).$$

The corresponding algorithm is as follows ([5, p.28]):

INPUT: $f, f_1, \ldots, f_s \in k[x_1, ..., x_n]$ with $f_i \neq 0$ $(1 \leq i \leq s)$
OUTPUT: $a_1, \ldots, a_s, r$ such that $f = a_1 f_1 + \ldots + a_s f_s + r$ and r is reduced with respect to $\{f_1, \ldots, f_s\}$ and $\max(LP(a_1)LP(f_1), \ldots, LP(a_s)LP(f_s), LP(r)) = LP(f)$.
INITIALIZATION: $a_1 := 0$, $a_2 := 0$, $\ldots$, $a_s := 0$, $r := 0$, $h := f$
WHILE $h \neq 0$ **DO**
IF there exists i such that $LT(f_i)$ divides $LT(h)$ **THEN** (Division Step)
choose the least i such that $LT(f_i)$ divides $LT(h)$

$$a_i := a_i + \frac{LT(h)}{LT(f_i)},\ h := h - \frac{LT(h)}{LT(f_i)} f_i.$$

ELSE (Remainder Step)

$$r := r + LT(h),\ h := h - LT(h)$$

Proof. To proof the existence of $a_1, \ldots, a_s$ and r we will show that the given algorithm operates correctly for any given input of polynomials. Fix a monomial order $>$ on $\mathbb{Z}^n_{\geq 0}$ and let $F = (f_1, \ldots, f_s)$ be an ordered s-tuple of polynomials in $k[x_1, ..., x_n]$. Pick $f \in k[x_1, ..., x_n]$. We will first show that in every stage of the algorithm it holds that

$$f = a_1 f_1 + \ldots + a_s f_s + h + r, \tag{3}$$

where $a_i, r \in k[x_1, ..., x_n]$ as in Theorem 2.11. We see that this is true for the initial values $a_1 := 0$, $a_2 := 0$, $\ldots$, $a_s := 0$, $r := 0$ and $h := f$. Assume now that (3) holds at one step of the algorithm. If a Division Step follows (some $LT(f_i)$ divides $LT(h)$), we redefine a_i and h. In this step $a_i f_i + h$ is unchanged:

$$a_i f_i + h = (a_i + \frac{LT(h)}{LT(f_i)}) f_i + h - (\frac{LT(h)}{LT(f_i)}) f_i.$$

Since only f_i is used, all other variables are unaffected so (3) remains true.
If no $LT(f_i)$ divides $LT(h)$ the Remainder Step takes place. Only r and h will be changed, but (3) preserves since their sum is unchanged:

$$h + r = (h - LT(h)) + (r + LT(h)).$$

We see that the algorithm stops when $h = 0$. In that case $f = a_1 f_1 + \ldots + a_s f_s + r$. In the previous steps we added terms to r when they were not divisible by any of the $LT(f_i)$. Therefore it follows that, when the algorithm terminates, $a_1, \ldots, a_s$ and r have the right properties as in Theorem 2.11.
Now it remains to show that the algorithm does eventually terminate. Crucial claim here is that in each step, h decreases in multidegree (relative to the term ordering). This is clear for a Remainder Step since its leading term is subtracted. In a Division Step we see that h is redefined as

$$h' := h - \frac{LT(h)}{LT(f_i)} f_i$$

By the first result of Lemma 2.8 we know that

$$LT(\frac{LT(h)}{LT(f_i)} f_i) = \frac{LT(h)}{LT(f_i)} LT(f_i) = LT(h),$$

so h' must have a strictly smaller multidegree than h when $h' \neq 0$. We see that during the algorithm a decreasing sequence of multidegrees is generated. Since the well-ordering property we know that such a sequence must terminate, in this case for $h = 0$.
The second statement of Theorem 2.11 is that if $a_i f_i \neq 0$, then we have $multideg(f) \geq multideg(a_i f_i)$. By Lemma 2.8 we know that $multideg(a_i f_i) = multideg(a_i) + multideg(f_i)$. Since the definitions in the Division Step it yields that every term in a_i is equal to $\frac{LT(h)}{LT(f_i)}$ for some value of h. Hence, $multideg(a_i) = multideg(\frac{LT(h)}{LT(f_i)}) = multideg(LT(h)) - multideg(f_i))$ for some value of h. Since

we start with $h := f$ and we showed that the sequences of multidegrees of h decreases, we know that $multideg(h) \leq multideg(f)$. From the algorithm we know that if $a_i f_i \neq 0$ then $multideg(f_i) \leq multideg(f)$. We can conclude

$$\begin{aligned} multideg(a_i f_i) &= multideg(h) - multideg(f) + multideg(f_i) \\ &\leq 2multideg(f) - multideg(f) \\ &= multideg(f). \end{aligned}$$

□

2.3 Dickson's Lemma on Monomial Ideals

Definition 2.12. *([3, p.68]) An ideal $I \subset k[x_1, ..., x_n]$ is a* ***monomial ideal*** *if there is a subset $A \subset \mathbb{Z}^n_{\geq 0}$ (possibly infinite) such that I consists of all polynomials which are finite sums of the form $\sum_{\alpha \in A} h_\alpha x^\alpha$, where $h_\alpha \in k[x_1, ..., x_n]$. In this case, we write $I = \langle x^\alpha : \alpha \in A \rangle$.*

Example 2.13. *An example of a monomial ideal is $I = \langle x^5y^6, x^3y^4, xy^8 \rangle \subset k[x, y]$, where $A := \{(5, 6), (3, 4), (1, 8)\}$.*

Hence we see that a monomial ideal I is generated by a set of monomials and that its elements are polynomials. We started this chapter with the problem of determining whether a given polynomial is in the ideal I. Working towards an answer for this we first discuss the problem for monomials.
Note that x^β is divisible by x^α exactly when $x^\beta = x^\alpha \cdot x^\gamma$ for some $\gamma \in \mathbb{Z}^n_{\geq 0}$.

Lemma 2.14. *([3, p.69]) Let $I = \langle x^\alpha : \alpha \in A \rangle$ be a monomial ideal. Then a monomial x^β lies in I if and only if x^β is divisible by x^α for some $\alpha \in A$.*

Proof. ($\Rightarrow$) Suppose $x^\beta \in \langle x^\alpha : \alpha \in A \rangle$. By Definition 2.12, $x^\beta = \sum_{i=1}^{s} h_i x^{\alpha(i)}$, where $h_i \in k[x_1, ..., x_n]$ and $\alpha(i) \in A$. We can write h_i as linear combination of monomials with coefficients in k thus we know that every term in the left hand side is divisible by some $x^{\alpha(i)}$. Since x^β is the sum of these terms we conclude that x^β is divisible by some $x^{\alpha(i)}$ too.
($\Leftarrow$) Suppose x^β is divisible by x^α for some $\alpha \in A$. Then x^β is multiple of some $x^{\alpha(i)}$ with $\alpha(i) \in A$, therefore $x^\beta \in \langle x^\alpha : \alpha \in A \rangle = I$. □

We now show which are the characteristics of a polynomial f belonging to a monomial ideal.

Lemma 2.15. *([3, p.69]) Let I be a monomial ideal, and let $f \in k[x_1, ..., x_n]$. Then the following are equivalent:*

i. $f \in I$.

ii. *Every term of f lies in I.*

iii. *f is a linear combination of the monomials in I.*

Proof. The implications $(iii.) \Rightarrow (ii.) \Rightarrow (i.)$ are trivial since I is closed under addition. What remains is $(i.) \Rightarrow (iii.)$ so let us assume that $f \in I$. Suppose $I = \langle x^\alpha : \alpha \in A \rangle$, then $f = \sum_{i=1}^{s} h_i x^{\alpha(i)}$ where $h_i \in k[x_1, ..., x_n]$ and $\alpha(i) \in A$.

Since each h_i is a linear combination of monomials, we know that every term of f is divisible by some $x^{\alpha(i)}$, hence f is a linear combination of the monomials $x^{\alpha(i)}$ in I. □

Using the previous lemma's we can now formulate the main result of this section.

Theorem 2.16. *([3, p.70])* ***Dickson's Lemma***
A monomial ideal $I = \langle x^\alpha : \alpha \in A\rangle \subset k[x_1, ..., x_n]$ can be written in the form $I = \langle x^{\alpha(1)}, x^{\alpha(2)}, \dots, x^{\alpha(s)}\rangle$, where $\alpha(1), \alpha(2), \dots, \alpha(s) \in A$. In particular, I has a finite basis.

Proof. We will proof this theorem by induction on n, the number of variables. If $n = 1$, then $I \subset k[x_1]$ is generated by x_1^α with $\alpha \in A \subset \mathbb{Z}_{\geq 0}$. By definition of a monomial ordering we know that A has a smallest element, let say β. Then x^β divides all the other generators x_1^α and therefore generates the whole ideal: $I = \langle x^\beta\rangle$.
Now assume that $n > 1$ and that the theorem holds for $n - 1$. Thus we know that a monomial ideal $I = \langle x^\alpha : \alpha \in A\rangle \subset k[x_1, \dots, x_{n-1}]$ can be written in the form $I = \langle x^{\alpha(1)}, \dots, x^{\alpha(s)}\rangle$, where $\alpha(1), \dots, \alpha(s) \in A$. The variable we add will be denoted as y so that the monomials in $k[x_1, \dots, x_{n-1}, y]$ can be written as $x^\alpha y^m$, where $\alpha = (\alpha_1, \dots, \alpha_{n-1}) \in \mathbb{Z}^{n-1}_{\geq 0}$ and $m \in \mathbb{Z}_{\geq 0}$.
Suppose that $I \subset k[x_1, \dots, x_{n-1}, y]$ is a monomial ideal. In order to find the generators for I, let J be the ideal in $k[x_1, \dots, x_{n-1}]$ generated by the monomials x^α for which $x^\alpha y^m \in I$ for some $m \geq 0$. J is a monomial ideal since the α's for which $x^\alpha y^m \in I$ form a subset $A \subset \mathbb{Z}^{n-1}_{\geq 0}$ for which Definition 2.12 holds for J. By the inductive step we know that J has finitely many generators, namely $J = \langle x^{\alpha(1)}, \dots, x^{\alpha(s)}\rangle$.
The definition of J tells us now that for i between 1 and s, $x^{\alpha(i)}y^{m_i} \in I$ for some $m_i \geq 0$. Let m be the largest of m_i. Then, for each k between 0 and $m - 1$, we define the ideal $J_k \subset k[x_1, \dots, x_{n-1}]$ generated by the monomials x^β such that $x^\beta y^k \in I$. Hence, J_k is the part of I generated by the monomials containing y exactly to the power k. Using the inductive step again, we know that J_k is finitely generated by monomials: $J_k = \langle x^{\alpha_k(1)}, \dots, x^{\alpha_k(s_k)}\rangle$.
We now claim that I is generated by the following monomials:

$$\begin{array}{lll}
\text{from} \quad J & : & x^{\alpha(1)}y^m, \dots, x^{\alpha(s)}y^m, \\
\text{from} \quad J_0 & : & x^{\alpha_0(1)}, \dots, x^{\alpha_0(s_0)}, \\
\text{from} \quad J_1 & : & x^{\alpha_1(1)}y, \dots, x^{\alpha_1(s_1)}y, \\
 & \vdots & \\
\text{from} \quad J_{m-1} & : & x^{\alpha_{m-1}(1)}y^{m-1}, \dots, x^{\alpha_{m-1}(s_{m-1})}y^{m-1}.
\end{array}$$

To proof the claim, let $x^\beta y^p \in I$ with either $p \geq m$ of $p \leq m - 1$. If $p \geq m$, the argument uses the definition of J which says that J contains all monomials $x^{\alpha(i)}$ such that $x^{\alpha(i)}y^m \in I$ for some $m \geq 0$. Thus $x^\beta \in J$ and by Lemma 2.14 we know that $x^\beta y^p$ is divisible by some $x^{\alpha(i)}y^m$. Therefore, in the case $p \geq m$, the monomials $x^{\alpha(i)}y^m$ generate the elements $x^\beta y^p \in I$.
On the other hand, if $p \leq m-1$, we use the definition of J_k: for each k between 0 and $m-1$, we define the ideal $J_k \subset k[x_1, \dots, x_{n-1}]$ generated by the monomials

x^β such that $x^\beta y^k \in I$. Now, $x^\beta \in J_k$ for some k and therefore $x^\beta y^p$ for $p \leq m-1$ is divisible by some $x^{\alpha_p(j)} y^p$ by Lemma 2.14. We now see that the listed monomials generate an ideal having the same monomials as in I. By part (iii.) of Lemma 2.15 we know that a monomial ideal is uniquely determined by its monomials. Therefore we conclude that the monomial ideal generates by this set of monomials is exactly I.
We now complete the proof and therefore switch to the notation $x_n = y$. We need to show now that $I = \langle x^\alpha : \alpha \in A \rangle \subset k[x_1, ..., x_n]$ is generated by finitely many of these x^α's. The above list of monomials gave us a set of generators for I. This set is finite because there are $s_i \in \mathbb{Z}_{\geq 0}$ (finitely many) monomials in each row and there are m rows for some finite $m \in \mathbb{Z}_{\geq 0}$. Let us denote this finite set of generators by $\{x^{\beta(1)}, \ldots, x^{\beta(t)}\}$ for $x^{\beta(i)} \in I$. By Lemma 2.14 we know that $x^{\beta(i)}$ lies in I if and only if $x^{\beta(i)}$ is divisible by $x^{\alpha(i)}$ for some $\alpha(i) \in A$. Thus we replace each $x^{\beta(i)}$ by his divisor $x^{\alpha(i)}$ and we find $I = \langle x^{\alpha(1)}, \ldots, x^{\alpha(t)} \rangle$. □

Example 2.17. *Let us apply Dickson's Lemma to $I = \langle x^5y^6, x^3y^4, xy^8 \rangle \subset k[x_1, ..., x_n]$ as we considered in Example 2.13. We defined $J_k \subset k[x_1, \ldots, x_{n-1}]$ to be the ideal generated by the monomials x^β such that $x^\beta y^k \in I$. Here the maximal degree of y in I is 8. We find J_k for $k = 1, \ldots, 8$ to be*

$$J = \langle x \rangle,$$
$$J_0 = J_1 = J_2 = J_3 = \{0\},$$
$$J_4 = J_5 = \langle x^3 \rangle,$$
$$J_6 = J_7 = \langle x^5 \rangle.$$

By the proof of Dickson's Lemma, I generated by xy^8 (from J), x^3y^4 (from J_4), x^3y^5 (from J_5), x^5y^6 (from J_6) and x^5y^7 (from J_7). We conclude that I is finitely generated, namely $I = \langle xy^8, x^3y^4, x^3y^5, x^5y^6, x^5y^7 \rangle$.

3 Groebner Bases

3.1 Hilbert Basis Theorem

In the previous chapter we have proven that every monomial ideal in $k[x_1, ..., x_n]$ has a finite generating set. In this section we will prove that every ideal $I \subset k[x_1, ..., x_n]$ has a finite generating set. To do so, we will make use of the leading term of each $f \in I$, which is unique when we fix a monomial ordering. For any ideal $I \subset k[x_1, ..., x_n]$ we start with defining its ideal of leading terms as follows.

Definition 3.1. *[3, p.74] Let $I \subset k[x_1, ..., x_n]$ be an ideal other than 0.*

i. We denote by $LT(I)$ the set of leading terms of elements of I. Thus,

$$LT(f) = \{cx^\alpha : \text{there exists } f \in I \text{ with } LT(f) = cx^\alpha\};$$

ii. We denote by $\langle LT(I)\rangle$ the ideal generated by the elements of $LT(I)$.

Note that if we have a finite generated ideal, say $I = \langle f_1, \ldots, f_s\rangle$, then $\langle LT(f_1), \ldots, LT(f_s)\rangle$ and $\langle LT(I)\rangle$ may be different ideals. We know that $\langle LT(f_1), \ldots, LT(f_s)\rangle \subset \langle LT(I)\rangle$ since $LT(f_i) \in LT(I) \subset \langle LT(I)\rangle$. But the other inclusion is not always true, $\langle LT(I)\rangle$ can be strict larger. To see this, we consider an example.

Example 3.2. *([3, p.74]) Let $I = \langle f_1, f_2\rangle$, where $f_1 = x^3 - 2xy$ and $f_2 = x^2y - 2y^2 + x$ and we use the grlex ordering on monomials in $k[x, y]$. Then*

$$x \cdot (x^2y - 2y^2 + x) - y \cdot (x^3 - 2xy) = x^2, \text{ so } x^2 \in I.$$

Thus, $x^2 = LT(x^2) \in \langle LT(I)\rangle$. However, $x^2 \notin \langle LT(f_1), LT(f_2)\rangle$ since x^2 is not divisible by $LT(f_1) = x^3$ or $LT(f_2) = x^2y$ (Lemma 2.14).

In order to be able to use the theory from Chapter 2, we will now show that $\langle LT(I)\rangle$ is a monomial ideal. In the light of Dickson's Lemma, we will prove that it has a finitely generating set too.

Proposition 3.3. *([3, p.75]) Let $I \subset k[x_1, ..., x_n]$ be an ideal.*

i. $\langle LT(I)\rangle$ is a monomial ideal.

ii. There are $g_1, \ldots, g_t \in I$ such that $\langle LT(I)\rangle = \langle LT(g_1), \ldots, LT(g_t)\rangle$.

Proof. i. The leading monomials $LM(g)$ of elements $g \in I \setminus 0$ generate the monomial ideal $\langle LM(g) : g \in I \setminus 0\rangle$. Since $LT(f) = LC(f) \cdot LM(f)$ with $LC(f)$ a nonzero constant, we know that $\langle LM(g) : g \in I \setminus 0\rangle = \langle LT(g) : g \in I \setminus 0\rangle = \langle LT(I)\rangle$. Hence $\langle LT(I)\rangle$ is a monomial ideal.

ii. Since $\langle LT(I)\rangle$ is generated by $\{LM(g) : g \in I \setminus 0\}$, it follows from Theorem 2.16 (Dickson's Lemma) that $\langle LT(I)\rangle = \langle LM(g_1), \ldots, LM(g_t)\rangle$ for finitely many $g_1, \ldots, g_t \in I$. By the same reasoning as in the proof of the first part of the proposition, we now conclude that $\langle LT(I)\rangle = \langle LT(g_1), \ldots, LT(g_t)\rangle$. □

We are now ready to state and proof Hilbert Basis Theorem:

Theorem 3.4. *([3, p.75])* ***Hilbert Basis Theorem***
Every ideal $I \subset k[x_1, ..., x_n]$ has as finite generating set. That is, $I = \langle g_1, \dots, g_t \rangle$ for some $g_1, \dots, g_t \in I$.

Proof. If $I = \{0\}$, then the generating set is $\{0\}$ and we are done. If I contains some nonzero polynomial then by Proposition 3.3 we know that there are $g_1, \dots, g_t \in I$ such that $\langle LT(I) \rangle = \langle LT(g_1), \dots, LT(g_t) \rangle$. We will prove that $I = \langle g_1, \dots, g_t \rangle$.
($\supseteq$) $\langle g_1, \dots, g_t \rangle \subset I$, since each $g_i \in I$.
($\subseteq$) For this inclusion we need the division algorithm from Theorem 2.11, and therefore we fix a monomial ordering. Let $f \in I$ be any polynomial. If we apply the division algorithm to divide f by $g_1, \dots, g_t$, then we get

$$f = a_1 g_1 + \ldots + a_t g_t + r$$

where every term in r is not divisible by any of $LT(g_1), \dots, LT(g_t)$. Further, we need to show that $r = 0$. Since $f, g_1, \dots, g_t \in I$ then also $r = f - a_1 g_1 - \ldots - a_t g_t \in I$. What follows is $LT(r) \in \langle LT(I) \rangle = \langle g_1, \dots, g_t \rangle$. By Lemma 2.14 we know that $LT(r)$ is divisible by any of $LT(g_i)$. But if we assume $r \neq 0$, this is a contradiction to what we stated: "every term in r is not divisible by any of $LT(g_1), \dots, LT(g_t)$". We conclude that $r = 0$ and find

$$f = a_1 g_1 + \ldots + a_t g_t + 0 \in \langle g_1, \dots, g_t \rangle.$$

□

Before we define the Groebner Bases, we will consider a geometric consequence of Hilbert Basis Theorem. First, we recall the definition of an affine variety:

$$\mathbf{V}(f_1, \dots, f_s) = \{(a_1, \dots, a_n) \in k^n : f_i(a_1, \dots, a_n) = 0 \text{ for all } i\}.$$

Hence an affine variety is the set of solutions for a systems of polynomial equations. Since every ideal $I \subset k[x_1, ..., x_n]$ contains infinitely many polynomials, it makes sense to examine affine varieties defined by the ideal I:

Definition 3.5. *Let $I \subset k[x_1, ..., x_n]$ be an ideal. We will denote by $\boldsymbol{V}(I)$ the set:*

$$\boldsymbol{V}(I) = \{(a_1, \dots, a_n) \in k^n : f(a_1, \dots, a_n) = 0 \textit{ for all } f \in I\}.$$

The following proposition shows that the set $\mathbf{V}(I)$ is indeed an affine variety.

Proposition 3.6. *$\boldsymbol{V}(I)$ is an affine variety. In particular, if $I = \langle f_1, \dots, f_s \rangle$, then $\boldsymbol{V}(I) = \boldsymbol{V}(f_1, \dots, f_s)$.*

Proof. By Hilbert Basis Theorem we know that the ideal I has a finite generating set, let's say $I = \langle f_1, \dots, f_s \rangle$. The claim of this proposition is that $\mathbf{V}(I) = \mathbf{V}(f_1, \dots, f_s)$.
($\subseteq$) Since $f_i \in I$ and $f(a_1, \dots, a_n) = 0$ for all $f \in I$, then also $f_i(a_1, \dots, a_n) = 0$ for $i = 1, \dots, s$. Hence, $\mathbf{V}(I) \subseteq \mathbf{V}(f_1, \dots, f_s)$.
($\supseteq$) We know $(a_1, \dots, a_n) \in \mathbf{V}(f_1, \dots, f_s)$, $f \in I$ and $I = \langle f_1, \dots, f_s \rangle$. Since I is generated by $f_1, \dots, f_s$, we can write

$$f = \sum_{i=1}^{s} h_i f_i$$

for some $h_i \in k[x_1, ..., x_n]$. Computing $f(a_1, \ldots, a_n)$ we then get

$$f(a_1, \ldots, a_n) = \sum_{i=1}^{s} h_i(a_1, \ldots, a_n) f_i(a_1, \ldots, a_n) = \sum_{i=1}^{s} h_i(a_1, \ldots, a_n) \cdot 0 = 0.$$

We conclude that $\mathbf{V}(f_1, \ldots, f_s) \subseteq \mathbf{V}(I)$, therefore both varieties are equal. □

3.2 Properties and construction of Groebner bases

The basis $g_1, \ldots, g_t$ described in Theorem 3.4 has the property $\langle LT(I) \rangle = \langle LT(g_1), \ldots, LT(g_t) \rangle$. We have seen in Example 3.2 that not all bases of an ideal I have this property. Therefore, we give to these special bases a name.

Definition 3.7. *Fix a monomial order. A finite subset $G = \{g_1, \ldots, g_t\}$ of an ideal I is said to be a **Groebner basis** of I if*

$$\langle LT(g_1), \ldots, LT(g_t) \rangle = \langle LT(I) \rangle.$$

Proposition 3.8. *Fix a monomial order. Then every ideal $I \subset k[x_1, ..., x_n]$ other than $\{0\}$ has a Groebner basis. Furthermore, any Groebner basis of an ideal I is a basis of I.*

Proof. We know by Proposition 3.3 that there exists a set $G = \{g_1, \ldots, g_t\}$ with $g_i \in I$ such that $\langle LT(I) \rangle = \langle LT(g_1), \ldots, LT(g_t) \rangle$, which means now that G is a Groebner basis of I. The proof of Theorem 3.4 showed us that a set G with this property is a basis of I. □

We started Chapter 2 with the Ideal Membership Problem. Groebner bases turn out to have an answer for this problem. A very useful property namely is that the remainder of a polynomial in $k[x_1, ..., x_n]$ on division by a Groebner basis is unique. This means that reducing a polynomial (using the division algorithm in $k[x_1, ..., x_n]$) by the elements of a Groebner basis G will give a unique remainder.

Proposition 3.9. *Let $G=\{g_1, \ldots, g_t\}$ be a Groebner basis of an ideal $I \subset k[x_1, ..., x_n]$ and let $f \in k[x_1, ..., x_n]$. Then f can uniquely be written as*

$$f = g + r,$$

where $g \in I$ and no term of r is divisible by any $LT(g_i)$.

Proof. The division algorithm gives $a_1, \ldots, a_t, r$ such that $f = a_1g_1 + \ldots + a_tg_t + r$ where r is reduced with respect to $g_1, \ldots, g_t$. This means that no term of r is divisible by any $LT(g_i)$. Since G is a basis, $g = a_1g_1 + \ldots + a_tg_t \in I$. What remains is to show the uniqueness of r.
Assume the opposite, $f = g + r_1 = h + r_2$ with $g, h \in I$ and no term of r_1 and r_2 is divisible by any $LT(g_i)$. Then $r_2 - r_1 = g - h \in I$. Since $r_1 \neq r_2$ we know $LT(r_2 - r_1) \in \langle LT(I) \rangle = \langle LT(g_1), \ldots, LT(g_t) \rangle$. Then $LT(r_2 - r_1)$ is divisible by any $LT(g_i)$ by Lemma 2.14, but no term of r_1 and r_2 is divisible by any $LT(g_i)$. We conclude that $r_1 = r_2$ and therefore the remainder of a polynomial upon division by a Groebner basis is unique. □

What this proposition says is that if we have a Groebner basis $\{g_1, \ldots, g_t\}$ then dividing f by $\{g_1, \ldots, g_t\}$, or by $\{g_t, \ldots, g_1\}$ or by any other permutation of its elements, will give the same remainder. This brings us the following corollary concerning the Ideal Membership Problem:

Corollary 3.10. *Let $G = \{g_1, \ldots, g_t\}$ be a Groebner basis of an ideal $I \subset k[x_1, ..., x_n]$ and suppose $f \in k[x_1, ..., x_n]$. Then $f \in I$ if and only if the remainder on division of f by G is zero.*

Proof. Let $G = \{g_1, \ldots, g_t\}$ be a Groebner basis of an ideal $I \subset k[x_1, ..., x_n]$ and let $f \in k[x_1, ..., x_n]$. Then $f \in I$ if and only if f is a linear combination of $g_1, \ldots, g_t$ if and only if division of f by $G = \{g_1, \ldots, g_t\}$ gives remainder zero. □

From now on we will write

$$r = \overline{f}^G$$

for the remainder of f when divided by G.

We now have a test for ideal membership, which we can only use when we are given a Groebner basis. The question arises how to detect whether a given generating set is a Groebner basis. We will now work on such a test and present an algorithm for constructing a Groebner basis from a given basis of an ideal I.

Suppose we are given a generating set $\{f_1, \ldots, f_s\}$ for I. By Definition 3.7, if this set is not a Groebner basis then there must be an element in $\langle LT(I)\rangle$ that is not in $\langle LT(g_1), \ldots, LT(g_t)\rangle$. This is what we saw in Example 3.2, where the leading term of a linear combination of the f_i is not in $\langle LT(g_1), \ldots, LT(g_t)\rangle$. This can only happen when there are leading terms in such a combination cancelling, leaving only smaller terms. The next definition concerns also this cancellation of leading terms.

Definition 3.11. *Let $f, g \in k[x_1, ..., x_n]$ be nonzero polynomials.*

1. *If $multideg(f) = \alpha$ and $multideg(g) = \beta$, then let $\gamma = (\gamma_1, \ldots, \gamma_n)$, where $\gamma_i = \max(\alpha_i, \beta_i)$ for each i. We call x^γ the* ***least common multiple*** *of $LM(f)$ and $LM(g)$, so that we write $x^\gamma = LCM(LM(f), LM(g))$.*

2. *The* ***S-polynomial*** *of f and g is defined as*

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

Example 3.12. *Let $f = x_1x_2^2 - x_3$ and $g = x_2 - x_3^4$, and use lex order with $x_1 > x_2 > x_3$. Then $\gamma_1 = \max(1,0), \gamma_2 = \max(2,1), \gamma_3 = \max(0,0)$ thus $\gamma = (1,2,0)$ and*

$$\begin{aligned} S(f,g) &= \tfrac{x_1^1x_2^2}{x_1x_2^2} \cdot f - \tfrac{x_1^1x_2^2}{x_2} \cdot g \\ &= x_1x_2^2 - x_3 - x_1x_2 \cdot (x_2 - x_3^4) \\ &= x_1x_2x_3^4 - x_3. \end{aligned}$$

As illustrated by this example, S-polynomials are defined in such a way that cancellation of leading term takes place. We will now prove that every cancellation of leading terms among polynomials of the same multidegree results from an S-polynomial type of cancellation.

Lemma 3.13. *([5, p.40]) Suppose we have a sum $f = \sum_{i=1}^{s} c_i f_i$, where $c_i \in k$, $f_i \in k[x_1, ..., x_n]$ and multideg$(f_i) = \delta \in \mathbb{Z}^n_{\geq 0}$ for all i. If multideg$(f = \sum_{i=1}^{s} c_i f_i) < \delta$, then f is a linear combination of the S-polynomials $S(f_k, f_l)$ for $1 \leq k, l \leq s$, with coefficients in k. Furthermore, each $S(f_k, f_l)$ has multidegree $< \delta$.*

Proof. We know $f = \sum_{i=1}^{s} c_i f_i$, where $c_i \in k$, $f_i \in k[x_1, ..., x_n]$, $multideg(f_i) = \delta \in \mathbb{Z}^n_{\geq 0}$ for all i and $multideg(f) < \delta$. Suppose $LT(f_i) = LC(f_i)LM(f_i) = a_i x^\delta$ then $LC(c_i f_i) = c_i a_i$ and $LC(f) = \sum_{i=1}^{s} c_i a_i$ because every f_i has multidegree δ. Since $multideg(f) < \delta$ we know that its leading coefficient $c_1 a_1 + \ldots + c_s a_s$ equals zero. Also,

$$S(f_i, f_j) = \frac{x^\delta}{a_i x^\delta} \cdot f_i - \frac{x^\delta}{a_j x^\delta} \cdot f_j = \frac{1}{a_i} f_i - \frac{1}{a_j} f_j.$$

We see that the leading term of every $S(f_i, f_j)$ vanishes and therefore every $S(f_i, f_j)$ has multidegree less than δ. We now find

$$\begin{aligned} f &= c_1 f_1 + \ldots + c_s f_s \\ &= c_1 a_1 (\tfrac{1}{a_1} f_1) + \ldots + c_s a_s (\tfrac{1}{a_1} f_s), \end{aligned}$$

which we can write as the telescoping sum

$$\begin{aligned} f &= c_1 a_1 (\tfrac{1}{a_1} f_1 - \tfrac{1}{a_2} f_2) + (c_1 a_1 - c_2 a_2)(\tfrac{1}{a_2} f_2 - \tfrac{1}{a_3} f_3) + \ldots \\ &\quad + (c_1 a_1 + \ldots + c_{s-1} a_{s-1})(\tfrac{1}{a_{s-1}} f_{s-1} - \tfrac{1}{a_s} f_s) + (c_1 a_1 + \ldots + c_s a_s) \tfrac{1}{a_s} f_s \\ &= c_1 a_1 S(f_1, f_2) + (c_1 a_1 - c_2 a_2) S(f_2, f_3) + \ldots \\ &\quad + (c_1 a_1 + \ldots + c_{s-1} a_{s-1}) S(f_{s-1}, f_s) + 0 \cdot \tfrac{1}{a_s} f_s. \end{aligned}$$

We conclude that f is a linear combination of the S-polynomials $S(f_k, f_l)$ for $1 \leq k, l \leq s$, with coefficients in k. □

We now have all material needed to present the test for a generating set of an ideal to be a Groebner basis of that ideal.

Theorem 3.14. *([3, p.84])* ***Buchberger's S-Pair Criterion***
A basis $\{g_1, \ldots, g_t\} \subset I$ is a Groebner basis of I if and only if for all pairs $i < j$, we have

$$\overline{S(g_i, g_j)}^G = 0.$$

Proof. ($\Rightarrow$) If $G = \{g_1, \ldots, g_t\}$ is a Groebner basis of I, then since $S(g_i, g_j) \in I$, the remainder on division by G is zero by Corollary 3.10.
($\Leftarrow$) Assume $\overline{S(g_i, g_j)}^G = 0$ for all $i \neq j$. Let $f \in I$ be a nonzero polynomial. Because $I = \langle g_1, \ldots, g_t \rangle$ we know there are polynomials $h_i \in k[x_1, ..., x_n]$ such that $f = \sum_{i=1}^{t} h_i g_i$. From Lemma 2.8 we know that

$$multideg(f) \leq \max(multideg(h_i g_i)). \quad (4)$$

Let now $m(i) = multideg(h_i g_i)$ and $\delta := \max(m(1), m(2), \dots, m(s))$. Then we can write (4) as

$$multideg(f) \leq \delta.$$

Note that there are many different ways that f can be written as linear combination of the g_i's. For each sum we could possibly get a different δ. Since we make use of a monomial order (i.e. a well-ordering) we can select a combination for f with coefficients h_i such that δ is minimal.
We will show that once this δ is chosen, we have $multideg(f) = \delta$. This means an equality in (4), thus no cancellation of leading terms has occurred and hence $LT(f) \in \langle LT(g_1), \dots, LT(g_t)\rangle$. By Definition 3.7 we then have shown that $\{g_1, \dots, g_t\}$ is a Groebner basis of I.
We can write f as

$$\begin{aligned} f &= \textstyle\sum_{i=1}^{t} h_i g_i = \sum_{m(i)=\delta} h_i g_i + \sum_{m(i)<\delta} h_i g_i \\ &= \textstyle\sum_{m(i)=\delta} LT(h_i) g_i + \sum_{m(i)=\delta} (h_i - LT(h_i)) g_i + \sum_{m(i)<\delta} h_i g_i. \end{aligned}$$

In the second and third sum, the monomials have $multideg < \delta$. Our assumption $multideg(f) \leq \delta$ implies then that the first sum also has $multideg < \delta$.
Let $LT(h_i) = c_i x^{\alpha(i)}$. Then the first sum equals

$$\sum_{m(i)=\delta} LT(h_i) g_i = \sum_{m(i)=\delta} c_i x^{\alpha(i)} g_i = \sum_{m(i)=\delta} c_i f_i$$

with $f_i = x^{\alpha(i)} g_i$ and $multideg(f_i) = \delta$. Now we have the notation as in Lemma 3.13 thus f is a linear combination of the S-polynomials $S(x^{\alpha(k)} g_k, x^{\alpha(l)} g_l)$ for $1 \leq k, l \leq s$, with coefficients in k. About these S-polynomials we know

$$\begin{aligned} S(x^{\alpha(k)} g_k, x^{\alpha(l)} g_l) &= \tfrac{x^\delta}{x^{\alpha(k)} LT(g_k)} x^{\alpha(k)} g_k - \tfrac{x^\delta}{x^{\alpha(l)} LT(g_l)} x^{\alpha(l)} g_l \\ &= \tfrac{x^\delta}{LT(g_k)} g_k - \tfrac{x^\delta}{LT(g_l)} g_l \\ &= x^{\delta-\gamma_{kl}} (\tfrac{x^{\gamma_{kl}}}{LT(g_k)} g_k - \tfrac{x^{\gamma_{kl}}}{LT(g_l)} g_l) \\ &= x^{\delta-\gamma_{kl}} S(g_k, g_l), \end{aligned}$$

where $x^{\gamma_{kl}} = LCM(LM(g_k), LM(g_l))$. Hence there are constants $c_{kl} \in k$ such that

$$\sum_{m(i)=\delta} LT(h_i) g_i = \sum_{k,l} c_{kl} x^{\delta-\gamma_{kl}} S(g_k, g_l). \tag{5}$$

Since our assumption $\overline{S(g_i, g_j)}^G = 0$ for all $i \neq j$ we know that there are $a_{ikl} \in k[x_1, ..., x_n]$ such that

$$S(g_k, g_l) = \sum_{i=1}^{t} a_{ikl} g_i,$$

for which we know from the division algorithm that

$$multideg(a_{ikl} g_i) \leq multideg(S(g_k, g_l))$$

for all i, j, k. Multiplying $S(g_k, g_l)$ by $x^{\delta-\gamma_{kl}}$ we find

$$x^{\delta-\gamma_{kl}} S(g_k, g_l) = \sum_{i=1}^{t} b_{ikl} g_i, \tag{6}$$

for $b_{ikl} = x^{\delta-\gamma_{kl}} a_{ikl}$. Because $S(f_k, f_l)$ has multidegree $< \delta$, by Lemma 3.13 and by again applying the division algorithm, we have

$$multideg(b_{ikl} g_i) \leq multideg(x^{\delta-\gamma_{kl}} S(g_k, g_l)) < \delta. \tag{7}$$

Substituting (6) in (5) we get

$$\sum_{m(i)=\delta} LT(h_i) g_i = \sum_{k,l} c_{kl}(x^{\delta-\gamma_{kl}} S(g_k, g_l)) = \sum_{k,l} c_{kl}(\sum_i b_{ikl} g_i) = \sum_i \tilde{h_i} g_i. \tag{8}$$

By (7) we know that $multideg(\tilde{h_i} g_i) < \delta$.
Substituting (8) in 3.2, we find an expression for f where all terms have multidegree $< \delta$. This contradicts to the minimality of δ, thus we conclude $multideg(f) = \delta$ which completes the proof. □

Example 3.15. *[3, p.86] Consider the ideal $I = \langle y - x^2, z - x^3 \rangle$ of the twisted cubic in $\mathbb{R}^3$. The claim is that $G = \{y - x^2, z - x^3\}$ is a Groebner basis for lex order with $y > z > x$. To prove this, we can use Theorem* 3.14*. Because G has two elements here, we only have to check whether*

$$S(y - x^2, z - x^3) = \frac{yz}{y}(y - x^2) - \frac{yz}{z}(z - x^3) = -zx^2 + yx^3$$

reduces to zero after division by G. By the division algorithm we find

$$-zx^2 + yx^3 = x^3 \cdot (y - x^2) + (-x^2) \cdot (z - x^3),$$

thus $\overline{S(y - x^2, z - x^3)}^G = 0$. Thus now we have shown that G is indeed a Groebner basis of I.
When we use lex order with $x > y > z$, we see that G is no Groebner basis of $I = \langle -x^2 + y, -x^3 + z \rangle$. Namely,

$$S(-x^2 + y, -x^3 + z) = \frac{x^3}{-x^2}(-x^2 + y) - \frac{x^3}{-x^3}(-x^3 + z) = -xy + z,$$

which can not be reduced any further by G.

We see that the Buchberger's S-Pair Criterion not only gives a test for a generating set F to be a Groebner basis but also suggest a construction algorithm for a Groebner basis from a basis $F = \{f_1, \ldots, f_s\}$. By the Criterion, we namely want that for all pairs $i < j$ we have zero remainder $\overline{S(f_i, f_j)}^F$. When we encounter a nonzero remainder for a S-polynomial, we add this remainder to the set F, after which computing the same remainder will indeed give zero. Keep this 'new' F for reducing the other S-polynomials and again add nonzero remainders when they occur.
This results in the following algorithm due to Buchberger for computing a Groebner basis.

Theorem 3.16. *([2, p.32])* ***Buchberger's Algorithm*** *Given $f_1, \dots, f_s \subset k[x_1, ..., x_n]$ such that $I = \langle f_1, \dots, f_s \rangle$, consider the algorithm which starts with $F = \{f_1, \dots, f_s\}$ and then repeats the following steps:*

1. *(Compute Step) Compute $\overline{S(f_i, f_j)}^F$ for $f_i, f_j \in F$ with i, j such that $1 \leq i < j \leq s$.*
2. *(Augment Step) Augment F by adding the nonzero $\overline{S(f_i, f_j)}^F$.*

Repeat the steps until nothing is added to F in the Augment Step: all remainders are zero. The final value of F is a Groebner basis of the ideal I.

For details on the algorithm we refer the reader to [3, p.89] and [5, p.43].

Theorem 3.17. *([5, p.42]) Given $F = \{f_1, \dots, f_s\}$ with nonzero $f_i \in k[x_1, ..., x_n]$ for $1 \leq i \leq s$, Buchberger's Algorithm will terminate (in a finite number of steps) and will produce a Groebner basis of the ideal $I = \langle f_1, \dots, f_s \rangle$.*

Proof. We will first show by contradiction that the algorithm terminates in a finite number of steps. Assume that the algorithm does not terminate is a finite number of steps. We now consider what happens in each pass through both steps. Suppose we start with $F_1 := F$ and when the algorithm progresses, we construct $F_i \supseteq F_{i-1}$. We obtain a strictly increasing infinite sequence:

$$F_1 \subset F_2 \subset F_3 \subset \dots .$$

We obtain every F_i by adding a $h \in I$ to F_{i-1}. Here h is the nonzero remainder of division by F_{i-1} of a S-polynomial of two elements in F_{i-1}. The sequence is strictly increasing since we know that $h \notin F_{i-1}$. By the division algorithm we know that $LT(h) \notin \langle LT(F_{i-1}) \rangle = \langle f_1, \dots, f_t \rangle$. Thus we get again a strictly increasing infinite sequence:

$$LT(F_1) \subset LT(F_2) \subset LT(F_3) \subset \dots .$$

This is a contradiction to the Ascending Chain Condition (see [9, p.83]), thus we conclude that Buchberger's Algorithm terminates in a finite number of steps.

What remains to show is that, given $F = \{f_1, \dots, f_s\}$ with nonzero $f_i \in k[x_1, ..., x_n]$ for $1 \leq i \leq s$, the algorithm will produce a Groebner basis of the ideal $I = \langle f_1, \dots, f_s \rangle$. We denote the finally constructed F_i by G. First we need to show that G is a basis of I. We started with $F_1 := F$ which was a basis of I. During the algorithm we enlarged F_i to become F_{i+1} by adding a nonzero remainder h of an S-polynomial of two elements in F_i after division by F_i. Since both polynomials are in F_i, the S-polynomial is in F_i and by the division by $F_i \subset I$ we know that $F_{i+1} \subset I$. We know F_{i+1} (and eventually G) contains the basis F, hence it is a basis of I too. We now can use Theorem 3.14 which says that $G = \{g_1, \dots, g_t\} \subset I$ is a Groebner basis of I if and only if for all pairs $i < j$, we have

$$\overline{S(g_i, g_j)}^G = 0. \tag{9}$$

Since for $g_i, g_j \in G$ we have (9) by construction of G (which exists because the algorithm terminates), we know that G is a Groebner basis of the ideal I. □

Note that the Groebner bases constructed by Buchberger's Algorithm are often larger than necessary. This shows that Groebner bases built in this way might not be unique. In the following we will discuss how we can reduce a Groebner basis and eventually make it unique for a given ideal.

Lemma 3.18. *Let G be a Groebner basis of the polynomial ideal I. Let $p \in G$ be a polynomial such that $LT(p) \in \langle LT(G - \{p\})\rangle$. Then $G - \{p\}$ is also a Groebner basis of I.*

Proof. By definition of a Groebner basis we know $\langle LT(G)\rangle = \langle LT(I)\rangle$. If $LT(p) \in \langle LT(G - \{p\})\rangle$ then we know that $\langle LT(G - \{p\})\rangle = \langle LT(G)\rangle$. We conclude that $\langle LT(G - \{p\})\rangle = \langle LT(I)\rangle$ hence $G - \{p\}$ is a Groebner basis of I. □

After having removed all polynomials p with the property from Lemma 3.18 and additionally having made every leading coefficient 1, we have a Groebner basis which we will call minimal:

Definition 3.19. *A* ***minimal Groebner basis*** *for a polynomial ideal I is a Groebner basis G for I such that:*

i. $LC(p) = 1$ for all $p \in G$, and

ii. For all $p \in G$, $LT(p) \notin \langle LT(G - \{p\})\rangle$.

But still, a given ideal can have several distinct minimal Groebner bases. Namely, if we have a minimal Groebner basis, we can adjust it such that it remains to be minimal. For example, when $\{x^2, xy, y^2 - (1/2)x\}$ is a minimal Groebner basis of I then $\{x^2 + axy, xy, y^2 - (1/2)x\}$ is too (where $a \in k$ is a constant). To reach uniqueness for a Groebner basis, we select one such minimal basis.

Definition 3.20. *A* ***reduced Groebner basis*** *for a polynomial ideal I is a Groebner basis G for I such that:*

i. $LC(p) = 1$ for all $p \in G$;

ii. For all $p \in G$, no monomial of p lies in $\langle LT(G - \{p\})\rangle$.

Proposition 3.21. *([3, p.91]) Let $I \neq \{0\}$ be a polynomial ideal. Then, for a given monomial ordering, I has a unique reduced Groebner basis.*

Proof. Let G be a minimal Groebner basis of I. We say $g \in G$ is *reduced* for G provided that none of the monomials of g are in $\langle LT(G - \{g\})\rangle$.
Note that if g is reduced for G, then it is also reduced for any other minimal basis of I that contains g and has the same leading terms. Definition 3.20 involves only leading terms and no other monomials of its elements.
Suppose $g \in G$, let $g' := \overline{g}^{G-\{g\}}$ and $G' := (G - \{g\}) \cup \{g'\}$. This means that we replace g in G by its remainder on division by $G - \{g\}$. Our claim is now that G' is a minimal basis of I too. We know that $LT(g') = LT(\overline{g}^{G-\{g\}}) = LT(g)$ since the leading term of g does not vanish in g' because of part ii. in Definition 3.19. Since $\langle LT(G')\rangle = \langle LT(G)\rangle = \langle LT(I)\rangle$ (G is a Groebner basis of I) and G' is contained in I we know that G' is a Groebner basis. It is a minimal Groebner basis since the leading terms of its elements did not change with respect to G.

Also, by construction, g' is reduced for G'.
We can apply this process to all elements of G until all are reduced. It is known that once an element p is reduced for any minimal basis, it is also reduced for any other minimal basis of I containing p and having the same leading terms. Although the minimal Groebner basis G may change during some of those processes, we know that we will end up with a reduced Groebner basis.
What remains to be shown is uniqueness, what will be shown by contradiction. Assume that G and $\tilde{G}$ are both reduced Groebner bases for I. It can be shown that they have the same cardinality and moreover the same leading terms, i.e. $LT(G) = LT(\tilde{G})$ (for the proof we refer to Adams [5], Prop. 1.8.4, p.47). Thus given $g \in G$, there is a $\tilde{g} \in \tilde{G}$ such that $LT(g) = LT(\tilde{g})$. To work towards a contradiction, consider $g - \tilde{g}$. Since g and $\tilde{g}$ are both in I, this is in I, which is generated by G. Hence we know $\overline{g - \tilde{g}}^G = 0$ but also $LT(g) = LT(\tilde{g})$. Hence, these terms cancel in $g - \tilde{g}$, and the remaining terms are divisible by none of $LT(G) = LT(\tilde{G})$ since G and $\tilde{G}$ are reduced. It follows that $0 = \overline{g - \tilde{g}}^G = g - \tilde{g}$ and therefore $g = \tilde{g}$. We now conclude that $G = \tilde{G}$ which completes the proof. □

3.3 Elimination theory

To be able to use Groebner bases for integer linear programming, we need some results from elimination theory. A big part of integer programming is about solving a system of equations in, say, n variables. Elimination theory gives us two steps that help to solve such a system. The first is that we look at a subsystem of the complete system, where we have simpler equations with less variables (we *eliminate* variables). It is easier to find a solution for this simplified system than for the complete system. The second procedure is that we can *extend* these 'subsolutions' to solutions of the original system.

Definition 3.22. *[3, p.114] Given $I = \langle f_1, \ldots, f_s \rangle \subset k[x_1, ..., x_n]$, then the l-th* ***elimination ideal*** *I_l is the ideal of $k[x_{l+1}, \ldots, x_n]$ defined by*

$$I_l = I \cap k[x_{l+1}, \ldots, x_n].$$

Proposition 3.23. *The l-th elimination ideal I_l is an ideal of $k[x_{l+1}, \ldots, x_n]$.*

Proof. We have to show the three properties in Definition 1.6 for $I_l = I \cap k[x_{l+1}, \ldots, x_n]$ with I an ideal in $k[x_1, ..., x_n]$. First of all, $0 \in I_l$ since $0 \in I$ and $0 \in k[x_1, ..., x_n]$. Second, since I is closed under addition, and a sum of two polynomials in $x_{l+1}, \ldots, x_n$ is again a polynomial in these variables, we know that I_l is closed under addition too. Finally we have to show that for $f \in I_l$ and $h \in k[x_{l+1}, \ldots, x_n]$ it yields that $hf \in I_l$. Since I is an ideal we know that $hf \in I$, and also $hf \in k[x_{l+1}, \ldots, x_n]$, which completes the proof. □

It can now be shown that given an ideal I, elements of a Groebner basis of the ideal I only involving the variables $x_{l+1}, \ldots, x_n$ form the Groebner basis of the l-th elimination ideal I_l.

Theorem 3.24. *([3, p.114])* ***The Elimination Theorem***
Let $I \subset k[x_1, ..., x_n]$ be an ideal and let G be a Groebner basis of I with respect to lex order where $x_1 > x_2 > \ldots > x_n$. Then, for every $0 \leq l \leq n$, the set

$$G_l = G \cap k[x_{l+1}, \ldots, x_n]$$

is a Groebner basis of the l-th elimination ideal I_l.

Proof. Let $I \subset k[x_1, ..., x_n]$ be an ideal and G be a Groebner basis of I with respect to lex order. Since $G \subset I$ we know that for a fixed l between 0 and n, $G_l \subset I_l$. What remains to show (by definition of a Groebner basis) is

$$\langle LT(I_l)\rangle = \langle LT(G_l)\rangle.$$

Because $G_l \subset I_l$ one inclusion is obvious, thus we only need to show that for any $f \in I_l$ the leading term $LT(f)$ is divisible by $LT(g)$ for some $g \in G_l$.
Since $f \in I_l \subset I$ we know that $LT(f)$ is divisible by $LT(g)$ for some $g \in G$ since G is a Groebner basis of I. Because $f \in I_l$ we know that this $LT(g)$ only involves the variables $x_{l+1}, \ldots, x_n$. We use lex order where $x_1 > x_2 > \ldots > x_n$ thus every monomial involving $x_1, \ldots, x_l$ is greater than all monomials involving $x_{l+1}, \ldots, x_n$. This leads us to the fact that $LT(g) \in k[x_{l+1}, \ldots, x_n]$ implies $g \in k[x_{l+1}, \ldots, x_n]$. Hence for any $f \in I_l$ the leading term $LT(f)$ is indeed divisible by $LT(g)$ for some $g \in G_l$, which implies that G_l is a Groebner basis of I_l. □

In this case we used lex order, where first x_1 was eliminated, then x_2, then x_3 and so on. We can think of a more efficient monomial ordering when we want to eliminate *certain* variables. We can construct an elimination order in which we order the variables according to the order in which we want to eliminate them. In the next chapter we will discuss such 'compatible term orders'.
The basic idea is that we want to extend the partial solutions we obtain by applying the Elimination Theorem, so that we obtain a solution of the original system given by the integer linear program. But how should we do this and what happens for example when the original system does not have a solution? To illustrate the answer to these questions, we look again at the geometric consequence of looking for solutions to a system of equations.

Recall from Definition 3.5 that for an ideal $I \subset k[x_1, ..., x_n]$ the affine variety $\mathbf{V}(I)$ is defined as

$$\mathbf{V}(I) = \{(a_1, \ldots, a_n) \in k^n : f(a_1, \ldots, a_n) = 0 \text{ for all } f \in I.$$

Finding the points $(a_1, \ldots, a_n)$ of $\mathbf{V}(I)$ is equivalent to solving the system of equations $\{f(a_1, \ldots, a_n) = 0 \text{ for all } f \in I\}$. The basic idea is to build up these solutions one coordinate at a time. Fix some l with $0 \leq l \leq n$. Then we find the elimination ideal I_l, and corresponding affine variety $\mathbf{V}(I_l)$. The set of points $(a_{l+1}, \ldots, a_n) \in \mathbf{V}(I_l)$ is the set of partial solutions of the original system of equations, i.e. if there exists a solution for the original system then the final $n - l$ coordinates of such a point form a point in $\mathbf{V}(I_l)$. Extending the point $(a_{l+1}, \ldots, a_n)$ by one coordinate is equivalent to finding a_l such that $(a_l, a_{l+1}, \ldots, a_n) \in \mathbf{V}(I_{l-1})$. Since $a_{l+1}, \ldots, a_n$ are known, this means that we have to solve a set of univariate equations $f(x_l, a_{l+1}, \ldots, a_n) = 0$ for all $f \in I_{l-1}$. When we know that $I = \langle f_1, \ldots, f_r\rangle$, this is equivalent to solving a system of r univariate equations. While extending partial solutions to complete solutions, it can happen that some partial solutions do not extend to a complete solutions.

For example, see [3, p.116], consider the equations

$$\begin{aligned} xy &= 1, \\ xz &= 1. \end{aligned} \tag{10}$$

Here, $I = \langle xy-1, xz-1\rangle$. Applying the Elimination Theorem, we start with considering $I_1 = I \cap k[y,z]$. Eliminating x, we find that the both equations reduce to the single equation $z = y$, hence $I_1 = \langle y - z\rangle$. The partial solutions (y,z) are given by (a,a) for some constant a. Extending these solutions to complete solutions, we have to find x s.t.$(x,a,a) \in \mathbf{V}(I)$. Since now $I = \langle xa-1\rangle$, we find $x = \frac{1}{a}$ which implies that a can not be zero. This shows that the set of complete solutions is described by $(\frac{1}{a}, a, a)$ for constant $a \neq 0$. Hence $(0,0)$ is the only partial solution which does not extend to a complete solution.
To speed up the extension process, it would be nice if we could say in advance which partial solutions extend to complete solutions. Suppose we are at the moment when only the first variable x_1 is eliminated. We now want to determine whether it is possible to find a_1 such that $(a_1, a_2, \ldots, a_n) \in \mathbf{V}(I)$ while we already know that $(a_2, \ldots, a_n) \in \mathbf{V}(I_1)$. The Extension Theorem tells us when this can be done.

Theorem 3.25. *([3, p.163]) **The Extension Theorem** Let $I = \langle f_1, \ldots, f_s\rangle \subset \mathbb{C}[x_1, \ldots, x_n]$ and let I_1 be the first elimination ideal of I. For each $1 \leq i \leq s$, we can write f_i in the form*

$$f_i = g_i(x_2, \ldots, x_n)x_1^{N_i} + \textit{terms in which } x_1 \textit{ has degree smaller than } N_i,$$

where $N_i \geq 0$ and $g_i \in \mathbb{C}[x_2, \ldots, x_n]$ is nonzero. Suppose we have a partial solution $(a_2, \ldots, a_n) \in \boldsymbol{V}(I_1)$. If $(a_2, \ldots, a_n) \notin \boldsymbol{V}(g_1, \ldots, g_s)$, then there exists $a_1 \in k$ such that $(a_1, \ldots, a_n) \in \boldsymbol{V}(I)$.

3.4 Application to integer linear programming

We are now arriving at the point where Groebner bases and optimization will meet. This happens via a polynomial map between the polynomial rings $k[y_1, \ldots, y_m]$ and $k[x_1, ..., x_n]$.

Definition 3.26. *([5, p.79]) A k-algebra homomorphism is a ring homomorphism*

$$\phi : k[y_1, \ldots, y_m] \to k[x_1, ..., x_n]$$

which is also a k-vector space linear transformation. Such a map is uniquely determined by

$$\phi : y_i \mapsto f_i,$$

where $f_i \in k[x_1, ..., x_n]$ with $1 \leq i \leq m$.

That is, this map between polynomials is uniquely determined by its image on the variable y_i. Let $h \in k[y_1, \ldots, y_m]$, say $h = \sum_v c_v y_1^{v_1} y_2^{v_2} \cdots y_m^{v_m}$ where $c_v \in k$, $v = (v_1, \ldots, v_m) \in \mathbb{Z}^m_{\geq 0}$, and only finitely many c_v's are non-zero. Then we have

$$\begin{aligned}\phi(h) &= \phi(\textstyle\sum_v c_v y_1^{v_1} y_2^{v_2} \cdots y_m^{v_m}) = \sum_v \phi(c_v y_1^{v_1} y_2^{v_2} \cdots y_m^{v_m}) \\ &= \textstyle\sum_v c_v \phi(y_1)^{v_1} \phi(y_2)^{v_2} \cdots \phi(y_m)^{v_m} = \sum_v c_v f_1^{v_1} f_2^{v_2} \cdots f_m^{v_m} \\ &= h(f_1, \ldots, f_m) \in k[x_1, ..., x_n].\end{aligned}$$

Recall that the *kernel* of ϕ is the ideal

$$\ker(\phi) = \{h \in k[y_1, \ldots, y_m] \mid \phi(h) = 0\},$$

and the *image* of ϕ is

$$\mathrm{Im}(\phi) = \{f \in k[x_1, ..., x_n] \mid \exists h \in k[y_1, \ldots, y_m] \text{ with } f = \phi(h)\}$$

which is, by the definition of a ring homomorphism, a sub ring of $k[x_1, ..., x_n]$.

Theorem 3.27. *([9, p.126])* ***First Isomorphism Theorem*** *If $\phi : R \to S$ is a homomorphism of rings, then ϕ induces a isomorphism of rings*

$$R/\ker(\phi) \cong Im(\phi).$$

Our strategy now is the following:

1. Translate the integer linear programming minimization problem into a problem concerning polynomials;
2. Use the Groebner basis techniques described so far to solve the polynomial problem;
3. Make a backward translation from the solution of the polynomial problem into the solution of the integer linear programming minimization problem.

Recall from the first chapter that an integer linear programming minimization problem is to find $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m) \in \mathbb{N}^m$ such that

$$\begin{aligned} a_{11}\sigma_1 + a_{12}\sigma_2 + \ldots + a_{1m}\sigma_m &= b_1 \\ a_{21}\sigma_1 + a_{22}\sigma_2 + \ldots + a_{2m}\sigma_m &= b_2 \\ \vdots \qquad\qquad & \quad \vdots \\ a_{n1}\sigma_1 + a_{n2}\sigma_2 + \ldots + a_{nm}\sigma_m &= b_n \end{aligned} \tag{11}$$

which minimizes the 'cost function'

$$c(\sigma_1, \sigma_2, \ldots, \sigma_m) = \sum_{j=1}^{m} c_j \sigma_j. \tag{12}$$

Here, $a_{ij} \in \mathbb{Z}$, $b_i \in \mathbb{Z}$, and $c_j \in \mathbb{R}$, for $i = 1, \ldots, n$, $j = 1, \ldots, m$.
We will first focus on solving (11), for the case where $a_{ij}, b_j \geq 0$. To become a problem concerning polynomials, we assign a variable to each linear equation in (11). We denote these variables by $x_1, x_2, \ldots, x_n$. Each equation of the system is then represented as

$$x_i^{a_{i1}\sigma_1 + \ldots + a_{im}\sigma_m} = x_i^{b_i}, \text{ for } i = 1, \ldots, n.$$

Combining the n equalities we find that (11) can be rewritten as

$$x_1^{a_{11}\sigma_1+\ldots+a_{1m}\sigma_m}\cdots x_n^{a_{n1}\sigma_1+\ldots+a_{nm}\sigma_m}=x_1^{b_1}\cdots x_n^{b_n}$$

or equivalently,

$$(x_1^{a_{11}}x_2^{a_{21}}\cdots x_n^{a_{n1}})^{\sigma_1}\cdots(x_1^{a_{1m}}x_2^{a_{2m}}\cdots x_n^{a_{nm}})^{\sigma_m}=x_1^{b_1}\cdots x_n^{b_n}. \tag{13}$$

Consider now the following polynomial map

$$\begin{array}{rlcl}\phi: & k[y_1,\ldots,y_m] & \to & k[x_1,...,x_n]\\ & y_j & \mapsto & f_j=x_1^{a_{1j}}\cdots x_n^{a_{nj}}.\end{array} \tag{14}$$

We can now rewrite (13) as

$$(\phi(y_1))^{\sigma_1}\cdots(\phi(y_m))^{\sigma_m}=\phi(y_1^{\sigma_1}\cdots y_m^{\sigma_m})=x_1^{b_1}\cdots x_n^{b_n}.$$

We see that points in the feasible region of the integer linear minimization programming problem can be found by looking for power products in $k[y_1,\ldots,y_m]$ for which $x_1^{b_1}\cdots x_n^{b_n}$ is the image. This is stated in the next lemma, for which we have just illustrated the proof.

Lemma 3.28. *([5, p.106]) Assume that all a_{ij}'s and b_i's are non-negative. Then there exists a solution $(\sigma_1,\ldots,\sigma_m)\in\mathbb{Z}^m_{\geq 0}$ of the system* (11) *if and only if the power product $x_1^{b_1}\cdots x_n^{b_n}$ is the image under ϕ of a power product in $k[y_1,\ldots,y_m]$. Moreover, if $x_1^{b_1}\cdots x_n^{b_n}=\phi(y_1^{\sigma_1}\cdots y_m^{\sigma_m})$, then $(\sigma_1,\ldots,\sigma_m)$ is a solution of* (11)*.*

We are therefore interested in the image of ϕ. By the First Isomorphism Theorem we know that there is a connection with $\ker(\phi)$. Once we know what the kernel of ϕ is, we can more easily describe what the image of ϕ is and possibly also determine whether a given monomial is in the image.
Since the image of ϕ is exactly the set of polynomials in $k[x_1,...,x_n]$ that can be expressed as polynomials in the variables f_j for $j=1,\ldots,m$, we will denote the image by $k[f_1,\ldots,f_m]$.
The following theorem gives a way to find the elements of $\ker(\phi)$.

Theorem 3.29. *([5, p.81]) Let $\phi: k[y_1,\ldots,y_m]\to k[x_1,...,x_n]$ as in* (14) *and let $K=\langle y_1-f_1,\ldots,y_m-f_m\rangle\subseteq k[y_1,\ldots,y_m,x_1,\ldots,x_n]$ for $f_i\in k[x_1,...,x_n]$. Then* $\ker(\phi)=K\cap k[y_1,\ldots,y_m]$.

Proof. ($\subseteq$) Let $g\in\ker(\phi)\subset k[y_1,\ldots,y_m]$. Since $g\in k[y_1,\ldots,y_m]$, we can write

$$g=\sum_v c_v y_1^{v1}\cdots y_m^{v_m},$$

where $c_v\in k$, $v=(v_1,\ldots,v_m)\in\mathbb{N}^m$, and only finitely many c_v's are non-zero. Since $g\in\ker(\phi)$ we know that $\phi(g(y_1,\ldots,y_m))=g(f_1,\ldots,f_m)=0$. Therefore, we have

$$\begin{aligned}g&=g-g(f_1,\ldots,f_m)\\&=\textstyle\sum_v c_v y_1^{v1}\cdots y_m^{v_m}-\sum_v c_v f_1^{v1}\cdots f_m^{v_m}\\&=\textstyle\sum_v c_v(y_1^{v1}\cdots y_m^{v_m}-f_1^{v1}\cdots f_m^{v_m}).\end{aligned}$$

By Lemma 1.11 we know that each term in this sum is in K, and therefore $g \in K \cap k[y_1, \ldots, y_m]$.
($\supseteq$) Let $g \in K \cap k[y_1, \ldots, y_m]$. Since K is generated by the elements in $\{y_1 - f_1, \ldots, y_m - f_m\}$, we can write

$$g(y_1, \ldots, y_m) = \sum_{i=1}^{m} (y_i - f_i(x_1, \ldots, x_n))h_i,$$

where $h_i \in k[y_1, \ldots, y_m, x_1, \ldots, x_n]$. Therefore,

$$\phi(g) = g(f_1, \ldots, f_m) = \sum_{i=1}^{m} (f_i(x_1, \ldots, x_n) - f_i(x_1, \ldots, x_n))h_i = 0.$$

Hence $g \in \ker(\phi)$ and we finished the proof. □

Combining Theorem 3.29 and the Elimination Theorem we now know how to compute the Groebner basis (and therefore the elements) of $\ker(\phi)$. Namely, first compute a Groebner basis of the ideal $K = \langle y_1 - f_1, \ldots, y_m - f_m\rangle$ in $k[y_1, \ldots, y_m, x_1, \ldots, x_n]$ with respect to a lex ordering with $x_1 > x_2 > \cdots > x_n > y_1 > \cdots > y_m$. Then a Groebner basis of $K \cap k[y_1, \ldots, y_m]$ will be exactly the Groebner basis of K intersected with $k[y_1, \ldots, y_m]$ by the Elimination Theorem. Since $\ker(\phi) = K \cap k[y_1, \ldots, y_m]$ we then have a Groebner basis of $\ker(\phi)$.
We can now return to our main topic of determining whether the power product $x_1^{b_1} \cdots x_n^{b_n}$ is in the image of ϕ. For any $f \in k[x_1, ..., x_n]$ we know that $f \in \text{Im}(\phi)$ if and only if there exists $h \in k[y_1, \ldots, y_m]$ such that $f = \phi(h)$. From the First Isomorphism Theorem we know that $\phi(h) \cong h + \ker(\phi)$. Hence $f \in \text{Im}(\phi)$ if and only if f corresponds to some $h + \ker(\phi)$ for some $h \in k[y_1, \ldots, y_m]$. Thus when we have a Groebner basis $G = \{g_1, \ldots, g_s\}$ for $\ker(\phi)$, we know $f \in \text{Im}(\phi)$ if and only if $f = \sum_{i=1}^{s} p_i g_i$ for some $p_i \in k[y_1, \ldots, y_m]$. But $f = \sum_{i=1}^{s} p_i g_i$ if and only if $\overline{f}^G = h$, what will be stated and proved in the following theorem.

Theorem 3.30. *([5, p.82])*
Let $K = \langle y_1 - f_1, \ldots, y_m - f_m\rangle \subseteq k[y_1, \ldots, y_m, x_1, \ldots, x_n]$ be the ideal considered in Theorem 3.29, and let G be the reduced Groebner basis of K with respect to an elimination order with the x variables larger than the y variables. Then $f \in k[x_1, ..., x_n]$ is in the image of ϕ if and only if there exists $h \in k[y_1, \ldots, y_m]$ such that $\overline{f}^G = h$. In this case, $f = \phi(h) = h(f_1, \ldots, f_m)$.

Proof. ($\Rightarrow$) Suppose $f \in k[x_1, ..., x_n]$ is in the image of ϕ, so $f = \phi(h)$ for some $h \in k[y_1, \ldots, y_m]$. If we now substitute f_i for y_i, we see that $f = \phi(h(y_1, \ldots, y_m)) = h(f_1, \ldots, f_m)$. Consider now

$$f(x_1, \ldots, x_n) - h(y_1, \ldots, y_m) \in k[y_1, \ldots, y_m, x_1, \ldots, x_n].$$

Since $f(x_1, \ldots, x_n) - h(y_1, \ldots, y_m) = h(f_1, \ldots, f_m) - h(y_1, \ldots, y_m)$, we see that this polynomial has monomial pairs $f_1^{\alpha_1} \cdots f_m^{\alpha_m} - y_1^{\alpha_1} \cdots y_m^{\alpha_m}$. Then, by Lemma 1.11, $f(x_1, \ldots, x_n) - h(y_1, \ldots, y_m) \in K$. Let G be the reduced Groebner basis of K with respect to an elimination order with the x variables larger than the y variables. Then we know

$$\overline{f(x_1, \ldots, x_n) - h(y_1, \ldots, y_m)}^G = 0.$$

This implies $\overline{f(x_1,\dots,x_n)}^G = \overline{h(y_1,\dots,y_m)}^G$. By Proposition 3.9 we know that this remainder is unique. Let $\overline{f(x_1,\dots,x_n)}^G = \overline{h(y_1,\dots,y_m)}^G = m$. Since $h \in k[y_1,\dots,y_m]$, h can only be reduced by polynomials in G that have leading terms in y alone. Because we have an elimination order with the x variables larger than the y variables, we now know that these polynomials do not have terms with x variables, therefore they are in $k[y_1,\dots,y_m]$. Since these polynomials as well as h are in $k[y_1,\dots,y_m]$, we conclude that $m \in k[y_1,\dots,y_m]$, thus $\overline{f(x_1,\dots,x_n)}^G = m$ with $m \in k[y_1,\dots,y_m]$.
($\Leftarrow$) Suppose that for $f \in k[x_1,...,x_n]$ there exists $h \in k[y_1,\dots,y_m]$ such that $\overline{f}^G = h$, where G is the reduced Groebner basis of K. Then $\overline{f-h}^G = 0$ and therefore $f - h \in K$. By definition of K, we can write

$$f(x_1,\dots,x_n)-h(y_1,\dots,y_m) = \sum_{i=1}^{m} (y_i - f_i(x_1,\dots,x_n))q_i(y_1,\dots,y_m,x_1,\dots,x_n). \tag{15}$$

If we let homomorphism ϕ of (14) work on the right hand side of (15), we get

$$\begin{aligned}
&\phi(\textstyle\sum_{i=1}^{m} (y_i - f_i(x_1,\dots,x_n))q_i(y_1,\dots,y_m,x_1,\dots,x_n))\\
&= \textstyle\sum_{i=1}^{m} \phi((y_i - f_i(x_1,\dots,x_n))q_i(y_1,\dots,y_m,x_1,\dots,x_n))\\
&= \textstyle\sum_{i=1}^{m} (f_i(x_1,\dots,x_n) - f_i(x_1,\dots,x_n))q_i(f_1,\dots,f_m,x_1,\dots,x_n) = 0.
\end{aligned}$$

Therefore, using the left hand side we get

$$\begin{aligned}
0 &= \phi(f(x_1,\dots,x_n) - h(y_1,\dots,y_m)) = \phi(f(x_1,\dots,x_n)) - \phi(h(y_1,\dots,y_m))\\
&= f(x_1,\dots,x_n) - h(f_1,\dots,f_m).
\end{aligned}$$

We conclude that $f(x_1,\dots,x_n) = h(f_1,\dots,f_m) = \phi(h(y_1,\dots,y_m))$, hence f is in the image of ϕ. This completes the proof. $\square$

Suppose K and G as defined above. We can then combine Lemma 3.28, Theorem 3.29 and Theorem 3.30 by saying that when all a_{ij}'s and b_i's are non-negative, there exists a solution $(\sigma_1,\dots,\sigma_m) \in \mathbb{Z}^m_{\geq 0}$ of system (11) if and only if the power product $x_1^{b_1}\cdots x_n^{b_n}$ is the image under ϕ of $y_1^{\sigma_1}\cdots y_m^{\sigma_m}$. This is equivalent to the fact that for $y_1^{\sigma_1}\cdots y_m^{\sigma_m} \in k[y_1,\dots,y_m]$ it yields that $\overline{x_1^{b_1}\cdots x_n^{b_n}}^G = y_1^{\sigma_1}\cdots y_m^{\sigma_m}$.
This brings us to the following algorithm for determining whether an integer linear programming program has a feasible solution, and for finding a solution by using Groebner bases:

Groebner Algorithm for integer linear programming ([5, p.107])

1. Compute a Groebner basis G for $K = \langle y_j - x_1^{\alpha_{1j}} x_2^{\alpha_{2j}} \cdots x_n^{\alpha_{nj}} \mid j = 1, \ldots, m\rangle$ with respect to an elimination order with the x variables larger than the y variables;
2. Find the remainder h of the division of the power product $x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n}$ by G;
3. If $h \notin k[y_1, \ldots, y_m]$, then system (11) does not have non-negative integer solutions. If $h = y_1^{\sigma_1} \cdots y_m^{\sigma_m}$, then $(\sigma_1, \ldots, \sigma_m)$ is a solution of system (11).

Example 3.31. *([5, p.107]) We consider the system*

$$\begin{array}{rcrcrcrcl} 3\sigma_1 & + & 2\sigma_2 & + & \sigma_3 & + & \sigma_4 & = & 10 \\ 4\sigma_1 & + & \sigma_2 & + & \sigma_3 & & & = & 5. \end{array}$$

Introducing x_1, x_2, one for each equation, we find

$$\begin{array}{lcl} x_1^{3\sigma_1+2\sigma_2+\sigma_3+\sigma_4} & = & x_1^{10} \\ x_2^{4\sigma_1+\sigma_2+\sigma_3} & = & x_2^5. \end{array}$$

Combining and reordering both we get

$$(x_1^3 x_2^4)^{\sigma_1} (x_1^2 x_2)^{\sigma_2} (x_1 x_2)^{\sigma_3} (x_1)^{\sigma_4} = x_1^{10} x_2^5.$$

We have four y variables, one for each unknown. Then the corresponding polynomial map is

$$\begin{array}{lcl} \phi : \mathbb{Q}[y_1, y_2, y_3, y_4] & \to & k[x_2, x_2] \\ y_1 & \mapsto & x_1^3 x_2^4 \\ y_2 & \mapsto & x_1^2 x_2 \\ y_3 & \mapsto & x_1 x_2 \\ y_4 & \mapsto & x_1. \end{array}$$

Hence, $K = \langle y_1 - x_1^3 x_2^4, y_2 - x_1^2 x_2, y_3 - x_1 x_2, y_4 - x_1\rangle \subseteq \mathbb{Q}[y_1, y_2, y_3, y_4, x_1, x_2]$. By Buchberger's Algorithm it can be found that the Groebner basis of K with respect to the lex order with $x_1 > x_2 > y_1 > y_2 > y_3 > y_4$ is $G = \{f_1, f_2, f_3, f_4, f_5\}$ with

$$\begin{array}{rl} f_1 = & x_1 - y_4 \\ f_2 = & x_2 y_4 - y_3 \\ f_3 = & x_2 y_3^3 - y_1 \\ f_4 = & y_2 - y_3 y_4 \\ f_5 = & y_1 y_4 - y_4^3. \end{array}$$

Reducing $x_1^{10}x_2^5$ by f_1 and f_2 gives

$$\begin{aligned} x_1^{10}x_2^5 &\xrightarrow{f_1} x_1^9x_2^5y_4 \xrightarrow{f_1} x_1^8x_2^5y_4^2 \xrightarrow{f_1} \dots \xrightarrow{f_1} x_1x_2^5y_4^9 \xrightarrow{f_1} x_2^5y_4^{10} \\ &\xrightarrow{f_2} x_2^4y_4^9y_3 \xrightarrow{f_2} x_2^3y_4^8y_3^2 \xrightarrow{f_2} \dots \xrightarrow{f_2} y_4^5y_3^5, \end{aligned}$$

which corresponds to $(\sigma_1, \sigma_2, \sigma_3, \sigma_4) = (0, 0, 5, 5)$ being a solution for this system.

We are now able to find solutions for a system like (11). Note that when $x_1^{b_1}x_2^{b_2}\cdots x_n^{b_n}$ reduces to $y_1^{\sigma_1}y_2^{\sigma_2}\cdots y_m^{\sigma_m}$ by G, which can be reduced to $y_1^{\tau_1}y_2^{\tau_2}\cdots y_m^{\tau_m}$ by elements of G, then both $(\sigma_1, \dots, \sigma_m)$ and $(\tau_1, \dots, \tau_m)$ are solutions of the system $A\sigma = b$. To solve an integer linear programming minimization problem, we now want to find the solution that minimizes the cost function (12). This can be done by a clever choice of the elimination order. Until now, the only requirement on such an order is that the x variables have to be larger than the y variables. The idea is now to use a term order on the y variables which is related to the cost function.

Definition 3.32. *([5, p.110]) A term order $<_c$ on the y variables is said to be* ***compatible with the cost function c*** *and the map ϕ if*

$$\left.\begin{array}{c} \phi(y_1^{\sigma_1}y_2^{\sigma_2}\cdots y_m^{\sigma_m}) = \phi(y_1^{\sigma_1'}y_2^{\sigma_2'}\cdots y_m^{\sigma_m'}) \\ \text{and} \\ c(\sigma_1, \dots, \sigma_m) < c(\sigma_1', \dots, \sigma_m') \end{array}\right\} \Longrightarrow y_1^{\sigma_1}y_2^{\sigma_2}\cdots y_m^{\sigma_m} <_c y_1^{\sigma_1'}y_2^{\sigma_2'}\cdots y_m^{\sigma_m'}.$$

This means that we define an order $<_c$ on the y variables such that, when vectors σ and σ' are both solutions to the system (11) and σ gives a 'lower cost' than σ', then the monomial $y_1^{\sigma_1}y_2^{\sigma_2}\cdots y_m^{\sigma_m}$ is smaller with respect to $<_c$ than $y_1^{\sigma_1'}y_2^{\sigma_2'}\cdots y_m^{\sigma_m'}$. Such a term order will give us eventually the solutions to (11) with minimal cost.

Proposition 3.33. *([5, p.111]) We use notation set above. Let G be a Groebner basis of K with respect to an elimination order with the x variables larger than the y variables, and an order $<_c$ on the y variables which is compatible with the cost function c and the map ϕ. If $\overline{x_1^{b_1}x_2^{b_2}\cdots x_n^{b_n}}^G = y_1^{\sigma_1}y_2^{\sigma_2}\cdots y_m^{\sigma_m}$, where $y_1^{\sigma_1}y_2^{\sigma_2}\cdots y_m^{\sigma_m}$ is reduced with respect to G, then $(\sigma_1, \sigma_2, \dots, \sigma_m)$ is a solution of system (11) which minimizes the cost function c.*

Proof. Let $\overline{x_1^{b_1}x_2^{b_2}\cdots x_n^{b_n}}^G = y_1^{\sigma_1}\cdots y_m^{\sigma_m}$ with $y_1^{\sigma_1}\cdots y_m^{\sigma_m}$ reduced with respect to G. Then $(\sigma_1, \dots, \sigma_m)$ is a solution of system 11 by Lemma 3.28. To show that it minimizes the cost function c, assume to the contrary that there exists a solution $(\sigma_1', \dots, \sigma_m')$ of system 11 for which $\sum_{j=1}^m c_j\sigma_j' < \sum_{j=1}^m c_j\sigma_j$. By Definition 3.32 we then have that $y_1^{\sigma_1}\cdots y_m^{\sigma_m} >_c y_1^{\sigma_1'}\cdots y_m^{\sigma_m'}$ Note that $\phi(y_1^{\sigma_1}\cdots y_m^{\sigma_m}) = \phi(y_1^{\sigma_1'}\cdots y_m^{\sigma_m'})$, so that $y_1^{\sigma_1}\cdots y_m^{\sigma_m} - y_1^{\sigma_1'}\cdots y_m^{\sigma_m'} \in \ker(\phi)$. By Theorem 3.29 we know that $\ker(\phi) = K \cap k[y_1, \dots, y_m]$ and therefore $y_1^{\sigma_1}\cdots y_m^{\sigma_m} - y_1^{\sigma_1'}\cdots y_m^{\sigma_m'} \in K$. Since G is a Groebner basis of K, we know $\overline{y_1^{\sigma_1}\cdots y_m^{\sigma_m} - y_1^{\sigma_1'}\cdots y_m^{\sigma_m'}}^G = 0$. Since $y_1^{\sigma_1}\cdots y_m^{\sigma_m} >_c y_1^{\sigma_1'}\cdots y_m^{\sigma_m'}$ we have $LT(y_1^{\sigma_1}\cdots y_m^{\sigma_m} - y_1^{\sigma_1'}\cdots y_m^{\sigma_m'}) = y_1^{\sigma_1}\cdots y_m^{\sigma_m}$

which is reduced with respect to G. But then we encounter a contradiction since $y_1^{\sigma_1} \cdots y_m^{\sigma_m} - y_1^{\sigma'_1} \cdots y_m^{\sigma'_m}$ cannot reduce to 0 by G. $\square$

In the case of all c_j being nonnegative it is possible to define a *weight order* $>_c$ on the y variables using cost function-values:

$$\begin{aligned} &y_1^{\sigma_1} \cdots y_m^{\sigma_m} >_c y_1^{\sigma'_1} \cdots y_m^{\sigma'_m} \text{ if } c(\sigma_1, \dots, \sigma_m) > c(\sigma'_1, \dots, \sigma'_m) \\ &\text{or } c(\sigma_1, \dots, \sigma_m) = c(\sigma'_1, \dots, \sigma'_m) \text{ and } c(\sigma_1, \dots, \sigma_m) >_\gamma c(\sigma'_1, \dots, \sigma'_m) \end{aligned}$$

where $>_\gamma$ any other fixed monomial order on the y variables. By definition this is a term order that is compatible with the cost function c and the map ϕ. When there are some $c_j < 0$ this order fails to be a well ordering. Hence we transform c into $\tilde{c}$ where all $\tilde{c}_j \geq 0$. This can be done by using the following (non-standard) degree:

$$\begin{aligned} &\deg x_i := 1 && \text{for all } i = 1, \dots, m, \\ &\deg y_j := d_j = \textstyle\sum_{i=1}^m a_{ij} && \text{for all } j = 1, \dots, n. \end{aligned}$$

Thus d_j is the sum of elements in A_j, column j of the matrix A. This is always positive since we assumed that all $a_{ij} \geq 0$ and otherwise the system would be independent of A_j. Therefore, given c_j from cost function c and $\mu > 0$ sufficiently large, we can construct

$$\tilde{c} = (c_1, \dots, c_n) + \mu(d_1, \dots, d_n)$$

for which all entries are positive. Consider now the weight vectors u_1, u_2 of length $(m + n)$:

$$\begin{aligned} u_1 &= (1, \dots, 1, 0, \dots, 0) \\ u_2 &= (0, \dots, 0, \tilde{c_1}, \dots, \tilde{c_n}). \end{aligned}$$

Then we can define a weight order $>_{u_1,u_2,\gamma}$ by comparing u_1 weights first, then comparing u_2 weights if the u_1 weights are equal, and finally breaking ties with any monomial order $>_\gamma$. Because we use u_1 first, we order exactly as a monomial order with all x_i (weight 1) bigger as any y_j (weight 0). Then you order the y_j mutually with respect to u_2. To see why $>_{u_2}$ is a compatible ordering, note that for $\tilde{c}$ and any two solutions $\sigma, \tau \in \mathbb{Z}^m_{\geq 0}$ we have

$$\begin{aligned} d \cdot \sigma = \textstyle\sum_j d_j \sigma_j &= \textstyle\sum_j (\sum_i A_{ij}) \sigma_j = \sum_i (\sum_j A_{ij} \sigma_j) \\ &= \textstyle\sum_i b_i = \sum_i (\sum_j A_{ij} \tau_j) = d \cdot \tau. \end{aligned}$$

Therefore $c \cdot \sigma < c \cdot \tau$ is equivalent to $c \cdot \sigma + \mu d \cdot \sigma < c \cdot \tau + \mu d \cdot \tau$ which is equivalent to $\tilde{c} \cdot \sigma < \tilde{c} \cdot \tau$. By this, we have shown that when changing c into $\tilde{c}$ nothing changes with respect to the compatibility. For details, see [4, p.386] and [6].

Above theory and algorithm are valid for the case where all a_{ij}'s and b_i's are non-negative. It remains to consider the case where we have a_{ij}'s and b_i's being any nonzero integer. We can not use the same algebraic translation here, since negative scalars are not allowed to be exponents in ordinary polynomials. One way to solve this, is by using so called *Laurent polynomials* in the variables x_i and x_i^{-1}.

Definition 3.34. *([4, p.311]) Let $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}^n$ be an integer vector. The corresponding **Laurent monomial** in variables $x_1, \ldots, x_n$ is*

$$x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}.$$

In general, we have

$$x^\alpha \cdot x^\beta = x^{\alpha+\beta} \text{ and } x^\alpha \cdot x^{-\alpha} = 1$$

for all $\alpha, \beta \in \mathbb{Z}^n$. Finite linear combinations

$$f = \sum_{\alpha \in \mathbb{Z}^n} c_\alpha x^\alpha$$

*of Laurent monomials are called **Laurent polynomials**. The collection of all Laurent polynomials form a commutative ring $k[x_1^{\pm 1}, \ldots, x_n^{\pm 1}]$ with coefficients in k, under the evident sum and product operations.*

Another useful representation of the ring of Laurent polynomials is ([4, p.313])

$$\begin{aligned} k[x_1^{\pm 1}, \ldots, x_n^{\pm 1}] &\cong k[x_1, \ldots, x_n, t_1, \ldots, t_n]/\langle x_1 t_1 - 1, \ldots, x_n t_n - 1\rangle \\ &\cong k[x_1, \ldots, x_n, t]/\langle x_1 \cdots x_n t - 1\rangle. \end{aligned}$$

Intuitively, this means that we introduce a new variable t, satisfying $x_1 \cdots x_n t - 1 = 0$, so that formally t is the product of inverses of the x_i.
Using the Laurent polynomials, we can extend the use of Groebner basis for integer programming to the case where a_{ij} and b_i for $i = 1, \ldots, n$, $j = 1, \ldots, m$ are any nonzero integers. For details we refer to [4, p.388-391]. Further on, we will consider that all a_{ij}'s and b_i's are non-negative.

3.5 Algorithm and comment

We implemented Groebner Algorithm for integer linear programming from [5] in Maple. For the code we refer the reader to Appendix $A.1$.
For the Maple procedure we use the spelling Grobner since the build-in package is already called Groebner. Input data of the procedure *Grobner* are the matrix $A \in \mathbb{Z}_{\geq 0}^{n \times m}$, vector $b \in \mathbb{Z}_{\geq 0}^n$ and list $c1$ with the coefficients $c1_i \in \mathbb{Z}$ of the linear objective function $f(x) = c1^T x$.
In the first loop, the linear constraints are transformed into polynomial constraints, as described in the previous section. At the end we construct the polynomial ideal $K = \langle y_j - x_1^{\alpha_{1j}} \cdots x_n^{\alpha_{nj}} \mid j = 1, \ldots, m\rangle \subset k[x_1, \ldots, x_n, y_1, \ldots, y_m]$. We also construct a monomial RHS in the variables $x_1, \ldots, x_n$, representing the right hand side of the system of linear equations. As mentioned, when all $c1_j$'s are nonnegative, we use a $c := c1$ to define a weight order on the y variables. If there exists some j such that $c1_j < 0$, thus $\min(c1) < 0$, we use $c := \tilde{c1}$ to

define the weight order, with $\tilde{c1}_i = c_i + \mu d_i$. For the further process, we need that the x variables are larger than the y variables. That's why we use w as weight vector for the x_i, with $w_i = \max(c) + 1$ for all i.
Using this ordering, we compute Groebner basis G and compute remainder h of RHS after division by elements from G. As the algorithm from [5] says, if h does not lie in $k[y_1, \ldots, y_n]$ then the system does not have a nonnegative integer solution. In the end, we deduce the solution components $\sigma_i = \text{degree}(y_i)$ in h, which form our optimal solution σ. As output are printed σ, its objective value and the required CPU running time.

4 Graver Bases

4.1 Definition

In the previous chapter Groebner bases turned out to be very useful for integer linear (so linear objective and linear constraints) programming. But there are a lot of interesting problems which do not have a linear objective function, but do have linear constraints. To be able to solve these problems too, the so called Graver basis turns out to be a useful tool. They can be used in nonlinear integer programming, and are moreover able to find solutions with negative entries. We will start with some introductory terminology and theorems from [7, Ch.4].
We will study a special class of ideals in $k[x_1, ..., x_n]$. Fix a subset $\mathcal{A} = \{a_1, \ldots, a_n\} \subset \mathbb{Z}^d$. Each vector a_i is identified with a monomial

$$t^{a_i} = t_1^{a_{i,1}} t_2^{a_{i,2}} \cdots t_d^{a_{i,d}},$$

in the Laurent polynomial ring $k[t^{\pm 1}] = k[t_1^{\pm 1}, \ldots, t_d^{\pm 1}]$.

Definition 4.1. *i. A* ***semigroup*** *is a nonempty set G together with a binary operation on G which is associative: $a(bc) = (ab)c$ for all $a, b, c \in G$.*

ii. Let $(S, \star)$ and $(T, \circ)$ be semigroups. Then $\phi : S \to T$ is a ***semigroup homomorphism*** *if*

$$\phi(a \star b) = \phi(a) \circ \phi(b)$$

for all $a, b \in S$.

Consider now the semigroup homomorphism

$$\begin{array}{cccc} \pi : & \mathbb{N}^n & \to & \mathbb{Z}^d \\ & u = (u_1, \ldots, u_n) & \mapsto & u_1 a_1 + \ldots + u_n a_n. \end{array}$$

We see that

$$\pi(u) = u_1 \begin{pmatrix} a_{11} \\ \vdots \\ a_{1d} \end{pmatrix} + \ldots + u_n \begin{pmatrix} a_{n1} \\ \vdots \\ a_{nd} \end{pmatrix} = \begin{pmatrix} u_1 a_{11} + u_2 a_{21} + \ldots + u_n a_{n1} \\ u_1 a_{12} + u_2 a_{22} + \ldots + u_n a_{n2} \\ \vdots \\ u_1 a_{1d} + u_2 a_{2d} + \ldots + u_n a_{nd} \end{pmatrix} = A \cdot u$$

where the columns of A are $a_1, a_2, \ldots, a_n$. We often identify the set $\mathcal{A}$ and the matrix A. The integer linear minimization programming problem is now equivalent to finding a point in $\pi^{-1}(b)$ which minimizes the given cost function. The map π lifts to a homomorphism of semigroup algebras:

$$\begin{array}{cccc} \hat{\pi} : & k[x_1, ..., x_n] & \to & k[t^{\pm 1}] \\ & x_i & \mapsto & t^{a_i}. \end{array}$$

The homomorphisms π and $\hat{\pi}$ are related via

$$\hat{\pi}(x^u) = \hat{\pi}(x_1^{u_1} x_2^{u_2} \cdots x_n^{u_n}) = t^{a_1 u_1} t^{a_2 u_2} \cdots t^{a_n u_n} = t^{\pi(u)}.$$

The kernel of $\hat{\pi}$ is denoted as I_A and called the **toric ideal** of A. For this ideal the following holds:

Lemma 4.2. *([7, p.31]) The toric ideal I_A is spanned as k-vector space by the set of binomials*

$$\{x^u - x^v \mid u, v \in \mathbb{N}^n \text{ with } \pi(u) = \pi(v)\}.$$

Proof. A binomial $x^u - x^v$ lies in I_A if and only if $\hat{\pi}(x^u - x^v) = \hat{\pi}(x^u) - \hat{\pi}(x^v) = x^{\pi(u)} - x^{\pi(v)} = 0$. What remains to show is that every polynomial $f \in I_A$ is a linear combination of such binomials with coefficients in k. Fix a term order $<$ on $k[x_1, ..., x_n]$. Suppose $f \in I_A$ can not be written as such a linear combination. Choose f such that $LM_<(f) = x^u$ is smallest with respect to $<$ for all such polynomials. Since $f \in I_A$ we know

$$0 = \hat{\pi}(f) = f(t^{a_1} \cdots t^{a_n}) = t^{\pi(u)} + \text{other terms}.$$

In particular, we know that the term $t^{\pi(u)}$ must cancel. Therefore, there exists a monomial x^v in f, with $x^u > x^v$ such that $\pi(u) = \pi(v)$. We know that $f' = f - (x^u - x^v)$ cannot be written as a k-linear combination of binomials since otherwise f could. Since now $LM(f) < LM(f')$, we come to a contradiction for the minimality property of f. ☐

Definition 4.3. *We define the **support** of a vector u in $\mathbb{Z}^n$ to be*

$$supp(u) = \{i \mid u(i) \neq 0\}.$$

Corresponding to this we have that the support of a monomial $w \in k[x_1, ..., x_n]$ is $supp(w) = \{x_i \mid x_i \text{ divides } w\}$. The support of a binomial $f = u - v$ is $supp(f) = supp(u) \cup supp(v)$.

We can reformulate Lemma 4.2 now by using that every vector $u \in \mathbb{Z}^n$ can be written uniquely as $u = u^+ - u^-$ where u^+ and u^- are nonnegative and have disjoint support. This is because u^+ has entries $u^+(i) = \max(0, u(i))$ and $u^-(i) = \max(0, -u(i))$ for $i = 1, \ldots, n$.
We can extend the map π to the map

$$\begin{array}{cccc} \breve{\pi}: & \mathbb{Z}^n & \rightarrow & \mathbb{Z}^d \\ & v = v^+ - v^- & \mapsto & \pi(v^+) - \pi(v^-). \end{array}$$

As expected, when $v \in \mathbb{N}^n$, thus $v = v^+$, $\breve{\pi}$ is identical to π. We will write $\ker(\breve{\pi})$ for the set of nonzero vectors $u \in \mathbb{Z}^n$ for which $\pi(u^+) = \pi(u^-)$. This brings us to the following restatement of Lemma 4.2.

Corollary 4.4. *([7, p.32])*

$$I_A = \langle x^{u^+} - x^{u^-} \mid u \in \ker(\breve{\pi}) \rangle.$$

Note that the S-polynomial of two binomials (Definition 3.11) is either 0 or a binomial. Also, polynomial division of a binomial by a set of binomials is either 0 of a binomial. Therefore, Buchbergers's Algorithm on a set of binomials gives eventually a Groebner basis consisting of binomials. Since by Hilbert Basis Theorem this first set of binomials is finite, we come to the following result.

Corollary 4.5. *([7, p.32]) For every term order $<$, there is a finite set of vectors $\mathcal{G}_< \subset \ker(\breve{\pi})$ such that the reduced Groebner basis of I_A with respect to $<$ is $\{x^{u^+} - x^{u^-} \mid u \in \mathcal{G}_<\}$.*

We define the **universal Groebner basis** $\mathcal{U}_A$ to be the union of all reduced Groebner bases $\mathcal{G}_<$ of the toric ideal I_A as $<$ runs over all term orders. This is a Groebner basis of I_A with respect to any term order.

Definition 4.6. *A binomial $x^{u^+} - x^{u^-}$ in I_A is called a* ***circuit*** *if the support of u is minimal with respect to inclusion and the coordinates of $\boldsymbol{u}$ are relatively prime. We denote the set of circuits in I_A by $\mathcal{C}_A$.*

Equivalently, a circuit is a binomial $x^{u^+} - x^{u^-} \in I_A$ of two irreducible monomials, which has minimal support. This means there is no binomial $x^{v^+} - x^{v^-}$ such that $supp(v) \subset supp(u)$ and $supp(v) = supp(u)$, and such that $gcd(u_1, \ldots, u_n) = 1$.

Definition 4.7. *([7, p.33]) A binomial $x^{u^+} - x^{u^-}$ in I_A is called* ***primitive*** *if there exists no other binomial $x^{v^+} - x^{v^-}$ in I_A such that x^{v^+} divides x^{u^+} and x^{v^-} divides x^{u^-}. The set of all primitive binomials in I_A is called the* ***Graver basis*** *of A and written as Gr_A.*

In other words, a binomial $x^{u^+} - x^{u^-} \in I_A$ (or vector u) is called primitive if there exists no other binomial $x^{v^+} - x^{v^-} \in I_A$ (or vector v) such that both $u^+ - v^+$ and $u^- - v^-$ are nonnegative. Also, every primitive binomial consists of two irreducible monomials. It follows that every circuit is primitive, so that $\mathcal{C}_A \subseteq Gr_A$. In the following proposition we will show that the universal Groebner basis lies 'in between'.

Proposition 4.8. *For every finite set $A \subset \mathbb{Z}^n$, hence also for a matrix $A \in \mathbb{Z}^{n\times m}$ we have*

$$\mathcal{C}_A \subseteq \mathcal{U}_A \subseteq Gr_A.$$

Proof. ($\mathcal{C}_A \subseteq \mathcal{U}_A$) We will first show that every circuit in I_A lies in some reduced Groebner basis. Let $u \in \ker(\breve{\pi})$ be a circuit. Fix an elimination term order $>$ such that the x_i for which $u(i) = 0$ are lager than the x_j for which $u(i) \neq 0$, and $x^{u^+} > x^{u^-}$. To come to a contradiction, we assume that $x^{u^+} - x^{u^-}$ does not lie in the reduced Groebner basis of I_A. Then we know there exists a nonzero $v \in \ker(\breve{\pi})$ unequal to u, such that $x^{v^+} > v^{u^-}$ and the leading term x^{v^+} divides x^{u^+}. Then we know $supp(v^+) \subseteq supp(u^+) \subseteq supp(u)$. By the choice of term order this implies $supp(v^-) \subseteq supp(u)$, and hence $supp(v) \subseteq supp(u)$. Since u is a circuit, this is only possible for $u = v$. Contradiction, hence $x^{u^+} - x^{u^-} \in I_A$.
($\mathcal{U}_A \subseteq Gr_A$) We will now show that every binomial in $\mathcal{U}_A$ is primitive. Let $x^{u^+} - x^{u^-}$ be any binomial in the reduced Groebner basis $\mathcal{G}_<$ for I_A and let $u^+ > u^-$. Then x^{u^+} is a minimal generator of $LT(I_A)$ and x^{u^-} is not divisible by any of $LT(I_A)$ (by definition of a reduced Groebner basis). We are going to proof this inclusion by contradiction and therefore we assume that $x^{u^+} - x^{u^-}$ is not primitive. Then there exists some $v \in \ker(\breve{\pi})$ unequal to u such that x^{v^+} divides x^{u^+} and x^{v^-} divides x^{u^-}. If $v^+ > v^-$ then we have $u^+ > v^+ > v^-$ which contradicts to the fact that x^{u^+} is a minimal generator of $LT(I_A)$. If $v^- > v^+$ then we find $u^- > v^- > v^+$ which implies that x^{u^-} is divisible by $x^{v^+} \in LT(I_A)$, thus again a contradiction. $\square$

4.2 Connection between Groebner and Graver bases

In this section we will present the connection between Groebner and Graver bases, as described in [7, Ch.7]. The main result is Theorem 4.9, which leads to an algorithm for computing a Graver basis of a matrix A.
Consider the $(d+n) \times 2n$-matrix

$$\Lambda(A) = \begin{pmatrix} A & \mathbf{0} \\ I & I \end{pmatrix},$$

where I is the $n \times n$-identity matrix and $\mathbf{0}$ is the $d \times n$- zero matrix. The matrix $\Lambda(A)$ is called the **Lawrence lifting** of A and any matrix of this type is said to be of **Lawrence type**.
The matrices A and $\Lambda(A)$ have kernels that are isomorphic:

$$\ker(\Lambda(A)) = \{(u, -u) \mid u \in \ker(A)\}.$$

To see this, note that

$$\begin{pmatrix} A & \mathbf{0} \\ I & I \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} Au \\ u+v \end{pmatrix},$$

hence,

$$\begin{pmatrix} u \\ v \end{pmatrix} \in \ker(\Lambda(A)) \Leftrightarrow u \in \ker(A) \text{ and } v = -u.$$

The toric ideal $I_{\Lambda(A)}$ is the homogeneous prime ideal

$$I_{\Lambda(A)} = \langle x^{u^+} y^{u^-} - x^{u^-} y^{u^+} \mid u \in \ker(A) \rangle$$

in the polynomial ring $k[x_1, \ldots, x_n, y_1, \ldots, y_n]$. Clearly, $(u, -u)^+ = (u^+, u^-)$ and $(u, -u)^- = (u^-, u^+)$, which explains the exponents of the generators of $I_{\Lambda(A)}$.

Theorem 4.9. *([7, p.55]) For a Lawrence type matrix $\Lambda(A)$ the following sets of binomials coincide:*

i. the Graver basis of $\Lambda(A)$,

ii. the universal Groebner basis of $\Lambda(A)$,

iii. any reduced Groebner basis of $I_{\Lambda(A)}$,

iv. any minimal generating set of $I_{\Lambda(A)}$, up to multiples.

Proof. A vector $u \in \ker(A)$ is primitive if and only if the corresponding vector $(u, -u) \in \ker(\Lambda(A))$ is primitive. Hence, the Graver basis of $\Lambda(A)$ and the Graver basis of A are related as follows:

$$Gr_{\Lambda(A)} = \{x^{u^+} y^{u^-} - x^{u^-} y^{u^+} \mid x^{u^+} - x^{u^-} \in Gr_A\}.$$

Clearly, $Gr_{\Lambda(A)}$ is a generating set of $I_{\Lambda(A)}$, and it is a Groebner basis with respect to every term order. We must show that $Gr_{\Lambda(A)}$ is the unique minimal generating set of $I_{\Lambda(A)}$ (up to scalar multiples of the generators). Let $g := x^{u^+}y^{u^-} - x^{u^-}y^{u^+}$ be any element in $Gr_{\Lambda(A)}$ and let $B = Gr_{\Lambda(A)} \setminus \{g\}$. Suppose that B generates $I_{\Lambda(A)}$. Since $g \in I_{\Lambda(A)}$, we know g can be written as a polynomial linear combination of elements in B. That implies that there is a binomial $x^{v^+}y^{v^-} - x^{v^-}y^{v^+} \in B$ one of whose terms divides $LT(g) = x^{u^+}y^{u^-}$. This means that g is not primitive, which is a contradiction to the fact that $g \in Gr_{\Lambda(A)}$. We conclude that g, or some nonzero scalar multiple of g must appear in any minimal generating set of $I_{\Lambda(A)}$. □

4.3 Computing Graver bases

Theorem 4.9 in the previous section suggests the following algorithm for computing Graver bases:

Algorithm Graver basis:

1. Fix any term order on $k[x_1, \dots, x_n, y_1, \dots, y_n]$. Compute the reduced Groebner basis $\mathcal{G}$ of $I_{\Lambda(A)}$.
2. Substitute $y_1, \dots, y_n \mapsto 1$ in $\mathcal{G}$. The resulting subset of $k[x_1, ..., x_n]$ is the Graver basis Gr_A of the matrix A.

The correctness of the second step follows from the fact that

$$Gr_{\Lambda(A)} = \{x^{u^+}y^{u^-} - x^{u^-}y^{u^+} \mid x^{u^+} - x^{u^-} \in Gr_A\}.$$

Hence we see that by substituting $y_i = 1$ for all i in $Gr_{\Lambda(A)}$ we obtain the Graver basis Gr_A. By Dickson's Lemma we know that the Graver basis is always finite.

4.4 Alternative definition and equivalence

In [8] there is stated an alternative definition for a Graver basis. This definition differs at first sight from the definition in [7]. In this section we will discuss the alternative definition and in Proposition 4.14 we will show that both definitions are equivalent.

The **lattice** of A is $\mathcal{L}(A) := \{x \in \mathbb{Z}^n \mid Ax = 0\}$. The nonzero elements in this set are denoted by $\mathcal{L}^*(A) := \{x \in \mathbb{Z}^n \mid Ax = 0, x \neq 0\}$. Note that by definition $\ker(\breve{\pi}) = \mathcal{L}^*(A)$.
We define the partial order $\sqsubseteq$ on $\mathbb{R}^n$ as follows.

Definition 4.10. *For $x, y \in \mathbb{R}^n$ we write $x \sqsubseteq y$ (and say that x is conformal to y) if $x_i y_i \geq 0$ (x and y lie in the same orthant) and $|x_i| \leq |y_i|$ for $i = 1, \dots, n$. A finite sum $u := \sum_i v_i$ of vectors in $\mathbb{R}^n$ is conformal if $v_i \sqsubseteq u$ for all i and hence all summands lie in the same orthant.*

The alternative definition of a Graver basis now is

Definition 4.11. *([8, p.40]) The **Graver basis** of an integer matrix A is defined to be the finite set $Gr_A \subset \mathbb{Z}^n$ of $\sqsubseteq$-minimal elements in $\mathcal{L}^*(A)$.*

This set is finite because of the following extension of Gordan-Dickson Lemma ([10, p.44]):

Lemma 4.12. ***Gordan-Dickson Lemma, $\sqsubseteq$-version:***

1. *Every sequence $\{p_1, p_2, \ldots\}$ of points in $\mathbb{Z}^n$ such that $p_i \not\sqsubseteq p_j$ whenever $i < j$ is finite.*
2. *Every infinite set $S \subseteq \mathbb{Z}^n$ contains only finitely many $\sqsubseteq$-minimal points.*

A fundamental property of a Graver basis defined in this way is the following.

Lemma 4.13. *([8, p.41]) Every vector $x \in \mathcal{L}^*(A)$ is a conformal sum $x = \sum_i g_i$ of Graver basis elements $g_i \in Gr_A$, with some elements possibly appearing with repetitions.*

Proof. This can be proven by induction on $\sqsubseteq$, using that it is a well-ordering (i.e. Every non-empty subset of $\mathbb{Z}^n$ has a minimal element in this ordering). Consider any $x \in \mathcal{L}^*(A)$. If x is $\sqsubseteq$-minimal in $\mathcal{L}^*(A)$ then we are done. Assume for the induction step that all elements $z \in \mathcal{L}^*(A)$ such that $z \sqsubseteq y$ for some $y \in \mathcal{L}^*(A)$ can be written as a conformal sum. It remains to be shown that $x \in \mathcal{L}^*(A)$ with $y \sqsubset x$ can also be written as a conformal sum. There exists some element $g \in Gr_A$ such that $g \sqsubset x$ and $y = x - g$. By induction, $y = \sum_i g_i$ with $g_i \in Gr_A$ for all i. Now $x = g + y = g + \sum_i g_i$ is a conformal sum too.
Note that we can now write every element $x \in \mathcal{L}^*(A) = \ker(A)$ as a finite sign-compatible nonnegative integer linear combination $x = \sum_i \alpha_i g_i$ with $\alpha_i \in \mathbb{Z}_{\geq 0}$, $g_i \in Gr_A$ and $g_i \sqsubseteq x$ for all i. □

According to [8, p.42] only $i \leq 2n-2$ nonnegative integer coefficients α_i and Graver basis elements $g_i \in Gr(A)$ are needed.

Proposition 4.14. *Definition* 4.7 *and Definition* 4.11 *are equivalent.*

Proof. ($\Rightarrow$) Suppose we have $x^{u^+} - x^{u^-} \in Gr_A$, thus u represents a primitive binomial in I_A. We will use the notation $v \geq w$ for vectors $v, w \in \mathbb{Z}^m$ when $v_i \geq w_i$ for all i. By Definition 4.7 there exists no $x^{v^+} - x^{v^-} \neq x^{u^+} - x^{u^-}$ such that both $u^+ \geq v^+$ (1) and $u^- \geq v^-$ (2). We now know there exists no $v \in I_A$ with $v \neq u$ such that

$$|u_i| = \max(u_i^+, u_i^-) \geq \max(v_i^+, v_i^-) = |v_i|.$$

What remains to be shown is that there exists no $v \in I_A$ such that $u_i v_i \geq 0$ for all i. This follows immediately from inequalities (1) and (2). Namely, (1) implies that when u_i is nonpositive then v_i is too, and when v_i is positive then u_i is too. The implications of inequality (2) complete the proof. Conclusion is that u is $\sqsubseteq$-minimal in $\mathcal{L}^*(A)$. (Recall that by definition $x \sqsubseteq y$ if $x_i y_i \geq 0$ and $|x_i| \leq |y_i|$ for $i = 1, \ldots, n$.)
($\Leftarrow$) Suppose $\mathbf{u}$ is $\sqsubseteq$-minimal in $\mathcal{L}^*(A)$. Then we know that there exists no other vector $v \in \mathcal{L}^*(A)$ such that $u_i v_i \geq 0$ and $|v_i| < |u_i|$ for $i = 1, \ldots, n$. As mentioned before, the fact that $u_i v_i \geq 0$ means that u and v have elementwise the same sign. Using the fact that $|v_i| < |u_i|$, this implies both $v_i^+ < u_i^+$ and $v_i^- < u_i^-$. Hence there exists no other vector $v \in \ker(\breve{\pi})$ such that $u_i^+ - v_i^+$ and $u_i^- - v_i^-$ are both nonnegative. By Definition 4.7 we conclude that (u and therefore) $x^{u^+} - x^{u^-}$ is primitive in I_A. □

4.5 Application to Integer Programming

To see how Graver bases can be used in integer programming, we will discuss results from [10] and [8]. We consider the following integer minimization problem:

$$(\mathrm{IP})_{A,b,l,u,f}: \ \min\{f(x) \mid x \in \mathbb{Z}^m, Ax = b, l \leq x \leq u\},$$

with A an integer $n \times m$ matrix, $b \in \mathbb{Z}^n$, $l, u, \in \mathbb{Z}^m$ and $f : \mathbb{Z}^m \to \mathbb{Z}$ separable convex. We will denote the set of feasible points by

$$S = \{x \in \mathbb{Z}^m \mid Ax = b, l \leq x \leq u\}.$$

Definition 4.15. *([10, p.63]) A set $\mathcal{T} \subset \mathbb{Z}^m$ is called a* ***test set*** *or an* ***optimality certificate*** *for $(IP)_{A,b,l,u,f}$ if, for every nonoptimal feasible solution x_0 of $(IP)_{A,b,l,u,f}$, there exists a vector $t \in \mathcal{T}$ and some positive integer α such that*

i. $x_0 + \alpha t$ is feasible, and

ii. $f(x_0 + \alpha t) < f(x_0)$.

The vector $t \in \mathcal{T}$ (or αt) is called an ***improving direction****.*

Once we would have such a test set $\mathcal{T}$ that is finite, we see that there is some, so called, augmentation algorithm to find an optimal solution for $(IP)_{A,b,l,u,f}$. Namely, for a given feasible point x_0 we can check whether there are $t \in \mathcal{T}$ and $\alpha \in \mathbb{Z}_{\geq 0}$ with $x_0 + \alpha t$ feasible and $f(x_0 + \alpha t) < f(x_0)$. When this is the case, replace x_0 by $x_0 + \alpha t$ and repeat this procedure until we reach an optimal point. We will show in this section that especially the Graver basis Gr_A is such a finite test set. To do so, we first have to discuss the following lemma's.

Lemma 4.16. *([10, p.65]) Let x_0 and x_1 be feasible solutions to $Ax = b$, $l \leq z \leq u$. Moreover, let $x_1 - x_0 = \sum_i \alpha_i g_i$ be a nonnegative integer linear sign-compatible decomposition into Graver basis elements. If $\alpha_j > 0$ for some j, then for all β with $0 \leq \beta \leq \alpha_j$, the vector $x_0 + \beta g_j$ is also a feasible solution to $Ax = b$, $l \leq z \leq u$.*

Proof. Note that since x_0 and x_1 are both feasible and $g_i \in \ker(A)$, we know that $A(x_0 + \alpha_i g_i) = b$ and $x_1 - x_0 \in \ker(A)$. Therefore $x_1 - x_0 = \sum_i \alpha_i g_i$ is a sign-compatible (conformal) sum with $\alpha_i \in \mathbb{Z}_{\geq 0}$ and $g_i \in Gr_A$. This implies that all g_i that occur in the sum lie in the same orthant of $\mathbb{Z}^m$. Combining this with the fact that both

$$x_1 = x_0 + \sum_i \alpha_i g_i$$

and x_0 are bounded by l and u, we know that if $\alpha_j > 0$ for some j, then $x_1 = x_0 + \alpha_j g_j$ is also feasible. Namely, if there is a path of steps in the same orthant from one feasible point to another, then also a single step of those leads to a feasible point. Moreover, also a smaller step in this direction brings us to a point that is bounded by l and u, thus $x_0 + \beta_j g_j$ is feasible for all β with $0 \leq \beta \leq \alpha_j$. □

Lemma 4.17. *([8, p.43]) Let $f : \mathbb{R} \to \mathbb{R}$ be a univariate convex function, let r be a real number, and let $s_1, \ldots, s_n$ be real numbers satisfying $s_i s_j \geq 0$ for all i, j. Then we have the following:*

$$f(r + \sum_{i=1}^{m} s_i) - f(r) \geq \sum_{i=1}^{m} (f(r + s_i) - f(r)).$$

Lemma 4.18. *([8, p.44], [10, p.65]) Let $f : \mathbb{R}^m \to \mathbb{R}$ be any separable convex function, $x \in \mathbb{R}^m$ and $\sum \alpha_i g_i$ any conformal sum in $\mathbb{R}^m$ with $\alpha_i \in \mathbb{Z}_{\geq 0}$. Then we have the following inequality:*

$$f(x + \sum \alpha_i g_i) - f(x) \geq \sum \alpha_i (f(x + g_i) - f(x))$$

for arbitrary integers $\alpha_1, \ldots, \alpha_r$.

Proof. Let f_j be the univariate convex function such that $f(x) = \sum_{j=1}^{n} f_j(x_j)$. Suppose $a \leq j \leq n$. Since $\sum \alpha_i g_i$ is a conformal sum, we know that all occurring g_i lie in the same orthant, i.e. $g_{i,j} g_{k,j} \geq 0$ for all i, k. Set $r := x_j$ and $s_i := \alpha_i g_{i,j}$ and apply Lemma 4.17 to f_j. This gives

$$f_j(x_j + \sum \alpha_i g_{i,j}) - f_j(x_j) \geq \sum \alpha_i (f_j(x_j + g_{i,j}) - f_j(x_j)). \tag{16}$$

Summarizing the inequalities of (16) for $j = 1, \ldots, n$, we complete the proof. □

Lemma 4.19. *([10, p.65]) The set Gr_A is a test set for $(IP)_{A,b,l,u,f}$ for any vectors $b \in \mathbb{Z}^n$, $l, u \in \mathbb{Z}^m$ and for any separable convex function f given by a comparison oracle, that is, when queried on $x, y \in \mathbb{Z}^m$, it decides whether $f(x) < f(y)$, $f(x) = f(y)$, or $f(x) > f(y)$.*

Proof. Let x_0 be any feasible but nonoptimal solution of $(IP)_{A,b,l,u,f}$. Then there is some other feasible solution x_1 with $f(x_1) < f(x_0)$. Again, $x_1 - x_0 \in \ker(A)$ implies that

$$x_1 - x_0 = \sum_{i=1}^{r} \alpha_i g_i,$$

for some positive integers $\alpha_1, \ldots, \alpha_r$ and $g_i \in Gr_A$ such that $g_i \sqsubseteq x_1 - x_0$. By Lemma 4.16 we know that $x_0 + g_i$ is feasible for all i. Moreover, using Lemma 4.18, we know

$$\begin{aligned} 0 > f(x_1) - f(x_0) &= f(x_0 + (x_1 - x_0)) - f(x_0) \\ &= f(x_0 + \textstyle\sum_{i=1}^{r} \alpha_i g_i) - f(x_0) \\ &\geq \textstyle\sum_{i=1}^{r} \alpha_i (f(x_0 + g_i) - f(x_0)). \end{aligned}$$

Since α_i is positive for all i here, there must be at least one index i for which $f(x_0 + g_i) - f(x_0) < 0$. Hence, there exists a vector $g_i \in Gr_A$ and some positive integer α such that $x_0 + \alpha g_i$ is feasible and $f(x_0 + \alpha g_i) < f(x_0)$. By Definition 4.15, this completes the proof. □

Since the reduced Groebner basis of $I_{\Lambda(A)}$ is finite, we know that the Graver basis is finite too. Therefore, we have a finite test set for $(IP)_{A,b,l,u,f}$ which

leads to some augmentation algorithm, given an initial point. We will first focus on the initial point before discussing the Graver Algorithm.
The convergence of our algorithms will depend on the **binary length** of the input data. For example, the binary length of an integer $z \in \mathbb{Z}$ is the number of bits in its binary encoding, which is $\langle z \rangle := 1 + \lceil \log_2(|z|+1) \rceil$ with the extra bit for the sign. The binary length of a $A \in \mathbb{Z}^{n \times m}$, and in particular of a vector, is the sum $\langle A \rangle = \sum_{i,j} \langle a_{i,j} \rangle$ of lengths of its entries. The length of A is never smaller than the number of its entries, i.e. $\langle A \rangle \geq mn$. The length of a finite set of numbers, vectors of matrices, is the sum of lengths of its elements. An algorithm is **polynomial time** if its running time $T(n)$ is polynomial in the length of the input data, i.e. $T(n) = \mathcal{O}(n^k)$.
Let $\hat{f}$ denote the maximum value of $|f(x)|$ over the feasible set. This needs not to be part of the input. [8, p.11]

Lemma 4.20. *([8, p.42]) Let Gr_A be the Graver basis of matrix A and let $l, u \in \mathbb{Z}^m$. If there is some $g \in Gr_A$ satisfying $g_i \leq 0$ whenever $u_i < \infty$ and $g_i \geq 0$ whenever $l_i > -\infty$ then every set of the form $S := \{x \in \mathbb{Z}^m \mid Ax = b, l \leq x \leq u\}$ is either empty or infinite, whereas if there is no such g, then every set S of this form is finite. Clearly, the existence of such g can be checked in time polynomial in $\langle Gr_A, l, u \rangle$.*

Proof. First suppose there exists such $g \in Gr_A$ and consider any such S. Suppose S contains some point x. Then for all $\lambda \in \mathbb{Z}_{\geq 0}$, we have

$$l \leq x + \lambda g \leq u$$

since we walk away from the finite upper or lower boundary. Moreover, $A(x + \lambda g) = b$ and hence $x + \lambda g \in S$, hence S is infinite.
To proof the second part of the lemma, we assume that S is infinite. Then the polyhedron $P := \{x \in \mathbb{R}^m \mid Ax = b,\ l \leq x \leq u\}$ is unbounded. Thus it has a nonzero recession vector h, which we may assume to be integer, such that $x + \alpha h \in P$ for all $x \in P$ and $\alpha \geq 0$. This implies that $A(x + \alpha h) = b$ and therefore $h \in \ker(A)$. Moreover we know $l \leq x + \alpha h \leq u$ and thus $h_i \leq 0$ whenever $u_i < \infty$ and $h_i \geq 0$ whenever $l_i > -\infty$. By Lemma 4.13, h is a conformal (and therefore sign compatible) sum $h = \sum g_i$ of vectors $g_i \in Gr_A$. We conclude that each of these $g_i \in Gr_A$ also satisfies $g_i \leq 0$ whenever $u_i < \infty$ and $g_i \geq 0$ whenever $l_i > -\infty$. This existence proofs the seconds part of the lemma. $\square$

For the proof of the following Lemma and its algorithm, we refer to $[8, p.47]$.

Lemma 4.21. *([8, p.47]) There is an algorithm that, given $n \times m$ integer matrix A, its Graver basis Gr_A, $l, u \in \mathbb{Z}^m$ and $b \in \mathbb{Z}^n$, in time which is polynomial in $\langle A, Gr_A, l, u, b \rangle$, either finds a feasible point $x \in S$ or asserts that S is empty or infinite.*

Lemma 4.22. *([8, p.45]) There is an algorithm that, given integer $n \times m$ matrix A, its Graver basis Gr_A, vectors $l, u \in \mathbb{Z}^m$ and $x \in \mathbb{Z}^m$ with $l \leq x \leq u$, and separable convex function $f : \mathbb{Z}^m \to \mathbb{Z}$ presented by a comparison oracle, solves the integer program*

$$\min\{f(z) \mid z \in \mathbb{Z}^m,\ Az = b,\ l \leq z \leq u\},\ b := Ax,$$

in time which is polynomial in the binary length $\langle Gr_A, l, u, x, \hat{f} \rangle$ of the data.

Proof. First we can apply Lemma 4.20 to Gr_A and l, u and either determine that the feasible set S is infinite and stop, or conclude it is finite and continue. We stop when it is infinite because we are then unable to determine at which point the minimum takes place. Next, starting with $x_0 := x$ as initial point, produce a sequence of feasible points $x_0, x_1, \ldots, x_s$ by using the test set Gr_A. Having found x_k, we find x_{k+1} by solving for each $g \in Gr_A$ the following univariate minimization problem:

$$\min\{f(x_k + \lambda g) \mid \lambda \in \mathbb{Z}_{\geq 0},\ g \in Gr_A,\ l \leq x_k + \lambda g \leq u\}. \tag{17}$$

This can be done by defining S' to be the feasible set: $S' := \{\lambda \in \mathbb{Z}_{\geq 0}, l \leq x_k + \lambda g \leq u\}$. Determine $s := \sup(S')$. If $s = \infty$ we conclude that S' is infinite and stop. Otherwise, S' is a finite set of positive integers and the problem can be solved by a bisection method on the univariate function f. [8, p.44]
We now have first determined for each $g \in Gr_A$ the best λ to take, i.e. λ minimizes $f(x_k + \lambda g)$ with x_k and g fixed. After that, we selected the pair (λ, g) that corresponds to the smallest value $f(x_k + \lambda g)$. If $f(x_k + \lambda g) < f(x_k)$ then set $x_k := x_{k+1}$ and repeat. If not, then we have reached the optimum point, namely x_k. Since $g \in \ker(A)$ we know that every point in the sequence $x_0, x_1, \ldots, x_s$ is feasible. Since the feasible set is finite and the x_k have strict decreasing objective values, the algorithm terminates.
We now show that the final point x_s reached by the algorithm is optimal. Let x^* be and optimal solution to (17). Consider any point x_k in the sequence $x_0, x_1, \ldots, x_s$ and suppose it is not optimal. Since both x^k and x^* are feasible, we know that $x^* - x_k \in \ker(A)$ and by Lemma 4.13 it can be written as

$$x^* - x_k = \sum_{i=1}^{t} \lambda_i g_i,$$

involving $1 \leq t \leq 2n - 2$ elements $g_i \in Gr_A$ with coefficients $\lambda \in \mathbb{Z}_{>0}$. By Lemma 4.18 we can write

$$f(x^*) - f(x_k) = f(x_k + \sum_{i=1}^{t} \lambda_i g_i) - f(x_k) \geq \sum_{i=1}^{t} (f(x_k + \lambda_i g_i) - f(x_k)).$$

Adding $t(f(x_k) - f(x^*))$ to both sides we get

$$\begin{aligned}(t-1)(f(x_k) - f(x^*)) &\geq \textstyle\sum_{i=1}^{t} (f(x_k + \lambda_i g_i) - f(x_k)) + t(f(x_k) - f(x^*)) \\ &= \textstyle\sum_{i=1}^{t} (f(x_k + \lambda_i g_i)) - tf(x_k) + t(f(x_k) - f(x^*)) \\ &= \textstyle\sum_{i=1}^{t} (f(x_k + \lambda_i g_i) - f(x^*)).\end{aligned}$$

Hence, there is some summand on the right hand side that satisfies

$$f(x_k + \lambda_i g_i) - f(x^*) \leq \frac{t-1}{t}(f(x_k) - f(x^*)) \leq \frac{2n-3}{2n-2}(f(x_k) - f(x^*)).$$

The point $x_k + \lambda g$ attaining the minimum of (17) will satisfy $f(x_k + \lambda g) \leq f(x_k + \lambda_i g_i)$ for all i and therefore

$$f(x_k + \lambda g) - f(x^*) \leq \frac{2n-3}{2n-2}(f(x_k) - f(x^*)).$$

For the point $x_{k+1} = x_k + \lambda g$ now it holds that

$$f(x_{k+1}) - f(x^*) \leq \frac{2n-3}{2n-2}(f(x_k) - f(x^*)), \tag{18}$$

and therefore, $f(x_{k+1}) \leq f(x_k)$. This shows that the final point x_s produced by the algorithm is indeed optimal.
What remains to be shown is that the algorithm comes to an optimal point in polynomial time in the binary length of Gr_A, l, u, x and $\hat{f}$. In other words, we have to consider the boundary of the number of point s. Suppose x_s is the last point produced by the algorithm and consider any $i < s$ and the corresponding point $x_i \in \{x_0, x_1, \ldots, x_s\}$ which is nonoptimal. We know that $f(x_i) - f(x^*)$ is a positive integer, hence repeated use of (18) gives the following:

$$\begin{aligned} 1 \leq f(x_i) - f(x^*) &= \textstyle\prod_{k=0}^{i-1} \frac{f(x_{k+1})-f(x^*)}{f(x_k)-f(x^*)}(f(x) - f(x^*)) \\ &\leq (\tfrac{2n-3}{2n-2})^i (f(x) - f(x^*)). \end{aligned}$$

Applying log() on both sides we get

$$0 \leq i \log(\frac{2n-3}{2n-2}) + \log(f(x) - f(x^*)),$$

and therefore,

$$i \leq (\log(\frac{2n-2}{2n-3}))^{-1} \log(f(x) - f(x^*)).$$

This yields for all $i < s$ and hence, the number s of points produced by the algorithm is at most one unit larger than this bound. Looking at the Taylor series expansion at $x = 0$ we find $\log(x+1) \sim x$ and therefore

$$\log(\frac{2n-3}{2n-2}) \sim \frac{1}{2n-3}, \text{ hence } (\log(\frac{2n-2}{2n-3}))^{-1} \sim 2n-3.$$

We conclude

$$s = \mathcal{O}(n \log(f(x) - f(x^*))),$$

and thus the number of points produced and the total running time are polynomial. □

We are now able to formulate the main statement of this section, a result of [12], and restated by [8] as follows.

Theorem 4.23. *([12, Thm. 5b], [8, p.48]) There is an algorithm that, given integer $n \times m$ matrix A, its Graver basis Gr_A, $l, u \in \mathbb{Z}_\infty^m$, $b \in \mathbb{Z}^n$, and separable convex $f : \mathbb{Z}^n \to \mathbb{Z}$ presented by a comparison oracle, solves in time polynomial in $\langle A, Gr_A, l, u, b, \hat{f}\rangle$ the problem:*

$$\min\{f(x) \mid x \in \mathbb{Z}^m,\ Ax = b,\ l \leq x \leq u\}. \tag{19}$$

Proof. First apply the algorithm of Lemma 4.21 and conclude that for the given integer program the feasible set is empty or infinite and stop, or obtain an initial feasible solution and continue. Next, apply the algorithm of Lemma 4.22 and conclude that S' is infinite or obtain an optimal solution. As before, $\hat{f}$ needs not to be part of the input. □

4.6 Algorithm and comment

For the code of the following algorithms we refer the reader to Appendix A.2.

4.6.1 Initial point

Input data of the *Initial* algorithm are the matrix $A \in \mathbb{Z}_{\geq 0}^{m \times n}$, vector $b \in \mathbb{Z}_{\geq 0}^{m}$ and both boundary vectors $l, u \in \mathbb{Z}^n$. We first solve the system $Ax = b$ over the integers. We become a general solution which we subsequently restrict to the boundary vectors. Since the *isolve*-command can not handle inequalities, we use *LPSolve, assume=integer* to solve the system of inequalities over the integers. *LPSolve* needs an objective function too, which is set to the constant function 1. The Initial algorithm gives eventually an integer solution vector for the system $Ax = b,\ l \leq x \leq u$.

4.6.2 Graver basis

Input of the *Grbasis* algorithm is the matrix A. The code follows the algorithm introduced by Sturmfels ([7]). First, the Lawrence lifting *Alaw* of A is computed. By the use of Maple command *ToricIdealBasis*, we compute the reduced Groebner basis of the toric ideal defined by *Alaw*. Second, all y_i are set to 1 in this reduced Groebner basis. We now have the Graver basis of A, in the 'binomial'-form. For computational reasons we prefer the 'vector'-form and therefore we conclude this algorithm by a translation into vectors and letting them be the rows in the matrix G.

4.6.3 Graver main algorithm

Input data of this main *Graver* algorithm are the matrix $A \in \mathbb{Z}_{\geq 0}^{m \times n}$, vector $b \in \mathbb{Z}_{\geq 0}^{m}$, a separable convex function $f : \mathbb{Z}^n \rightarrow \mathbb{Z}$ and both boundary vectors $l, u \in \mathbb{Z}^n$. First the Graver basis of A is computed by using the *Grbasis* algorithm. Since Onn showed polynomial behavior 'given the Graver basis of A', we start the time measure after we've become the Graver basis of A. Then a feasible initial solution is computed, after which we apply the algorithm of Lemma 4.22. In the end we construct a matrix $K \in \mathbb{Z}^{p \times n}$ where the rows are the p feasible new points. We construct the list w of their objective values and select the smallest. If the objective value of this new point $x_{k+1} = x_k + \lambda_i g_i$ is strict smaller than the objective value of x_k then we replace x_k by x_{k+1}. Otherwise we found the optimal solution for (19) to be x_k. We conclude the algorithm with printing the optimal solution, its objective value and the CPU running time used for this algorithm.

5 Computational results

We saw in the previous chapter how we used results from both Sturmfels and Onn to come to the algorithms in Appendix $A.2$. The main result from Chapter 4, Theorem 4.23, was that these algorithms together solve the integer programming problem in (19) in time polynomial in $\langle A, Gr_A, l, u, b, \hat{f}\rangle$.
We now want to confirm this behavior by some computational examples. To do so, we applied the algorithm on 30 cases, each case for a linear function and for a sum of squares (which is separable convex). We denote each case by a number, we refer the reader to the appendices concerned to see which integer programming problem in meant by this number.
After comparing some computations, we found that we have to take into account a machine epsilon of about 0.4 seconds.

5.1 Groebner-Graver

First we compare the CPU time for the *Grobner* algorithm (Appendix $A.1$) and the *Graver* algorithm (Appendix $A.2$). Since *Grobner* only computes nonnegative solutions, we gave a zero vector to *Graver* as lower bound l. Upper bound u was taken large enough such that the optimal nonnegative solution would lie between l and u.
Here is the table of the time data we computed.

Table 1: Comparison Grobner/Graver

Case	CPU time *Grobner*	CPU time *Graver*
1.2	0.080	0.071
2.2	0.140	0.020
3.2	2.399	0.219
4.2	0.401	0.080
5a.2	0.610	0.030
5b.2	2105.650	0.100
5c.2	31.080	0.170
5d.2	188.320	2.180
5e.2	92.450	0.120
6.2	×	0.351
7.2	5.221	0.570
8.2	×	5.541
9.2	×	0.049
10.2	×	0.391

Here $\times$ means that the algorithm gave as output that in computing the Groebner basis Maple was unable to allocate enough memory to complete this computation. The precise data can be found in Appendix B.1. We see that in all 14 cases the *Graver* algorithm was faster with respect to the *Grobner* algorithm. It must be noted that the computation of the Graver basis in the *Graver* algorithm happens before the start of the time measure. This is because then Theorem 4.23 assures us that the bound on the running time will be polynomial in the binary length of some input data. This will not be the case when the computation of the Graver basis is included. "The Graver basis of an $m \times n$ matrix A is typically very large and cannot be written down, let alone computed, in polynomial time", [8, p.52]. Hence, when this computation is included in the time measurement, it is very likely that the *Grobner* algorithm will be faster in many cases.

5.2 Graver for linear objective functions

To study the behavior of our algorithm in this case, we have to compute $\langle A, Gr_A, l, u, b, \hat{f} \rangle$, where $\hat{f}$ denotes the maximum value of $|f(x)|$ over the feasible set described by using A, b, l, u. We've implemented an algorithm in which the feasible set is unknown at the beginning and is not computed during the algorithm either. For that reason, we can not determine $\hat{f}$ or the length of its binary representation. Therefore we computed $\langle A, Gr_A, l, u, b \rangle$ for each case to approach $\langle A, Gr_A, l, u, b, \hat{f} \rangle$. In the cases we computed, $\hat{f}$ is relatively small with respect to $\langle A, Gr_A, l, u, b \rangle$ and therefore $\langle A, Gr_A, l, u, b \rangle$ will be a good approximation. For example, suppose we have $\langle A, Gr_A, l, u, b \rangle = 14,348$ which could be the case for a problem with a 4×7 matrix A (look for example in Appendix B.2). If we now have the maximum of $|f(x)|$ over the feasible set is quite large, say $2,105,001$, then $\langle \hat{f} \rangle = 22$, which will affect the total binary length of the input data hardly. We will consider objective functions for which $\langle \hat{f} \rangle$ has this property.
For the data we refer the reader to the .2- cases in Appendix B.2. For the cases 1.2 until 10.2 now the lower bound has been redefined to be nonzero. Out of these data we can make the plot of Figure 1. On the x-axis we see the binary length $\langle A, Gr_A, l, u, b \rangle$, on the y-axis the CPU running time in seconds is displayed.

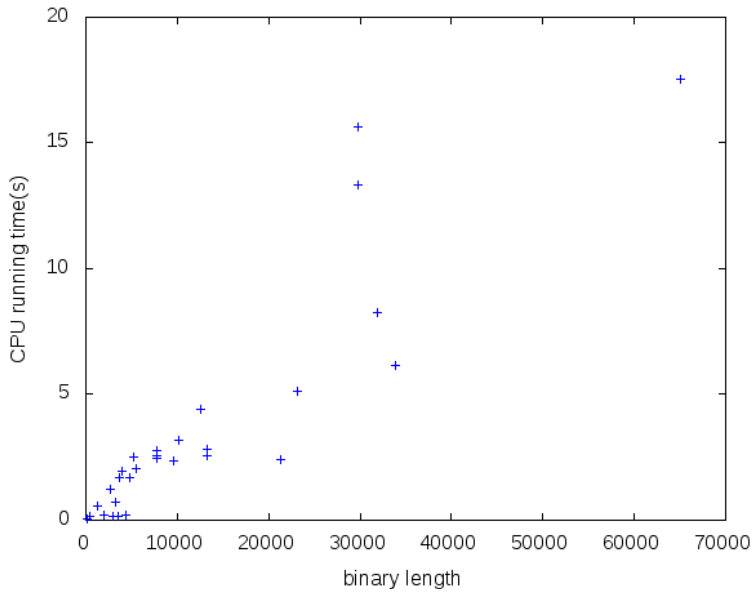

Figure 1: Plot for linear $f(x)$

Each $+$ denotes a measure point, there are in total 30 of them. Let now $T(l)$ denote the CPU running time and let l be the binary length $\langle A, Gr_A, l, u, b\rangle$. We can now write $T = cl^k$ for which we want to find constants $c, k \in \mathbb{R}$. To be able to determine these constants, we study the logarithm of these data. We see the plot in Figure 2. Since $T(l) = cl^k$, it holds that $\log(T) = \log(c) + k\log(l)$ for $T(l), l \geq 0$. We see in the figure that the logarithm of the data can be bounded by the line $\log(T) = -7.3 + \log(l)$, which is denoted by the red line. Comparing both equalities, we find $c = e^{-7.3}$ and $k = 1$.

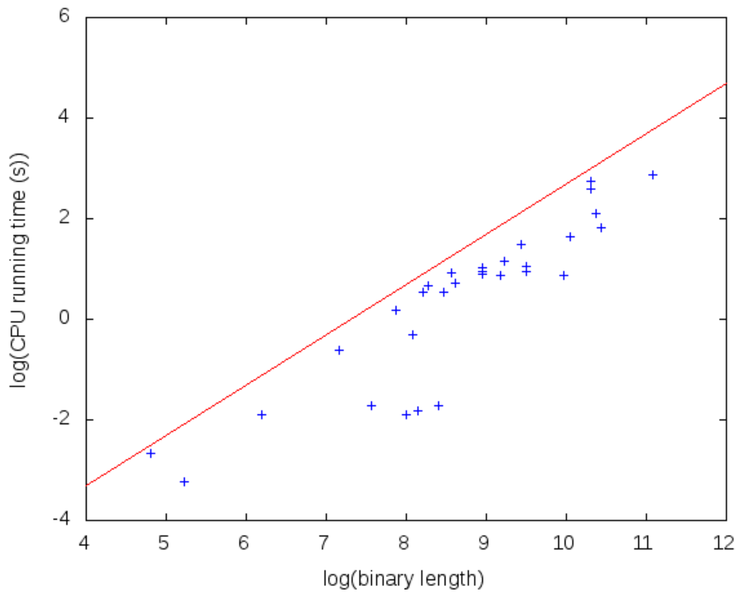

Figure 2: Log-plot for linear $f(x)$

Substituting these constants we find $T(l) = e^{-7.3}l$. We see in Figure 3 that this, denoted as the red line, is indeed the polynomial (in particular linear) bound on the running time for the computed data.

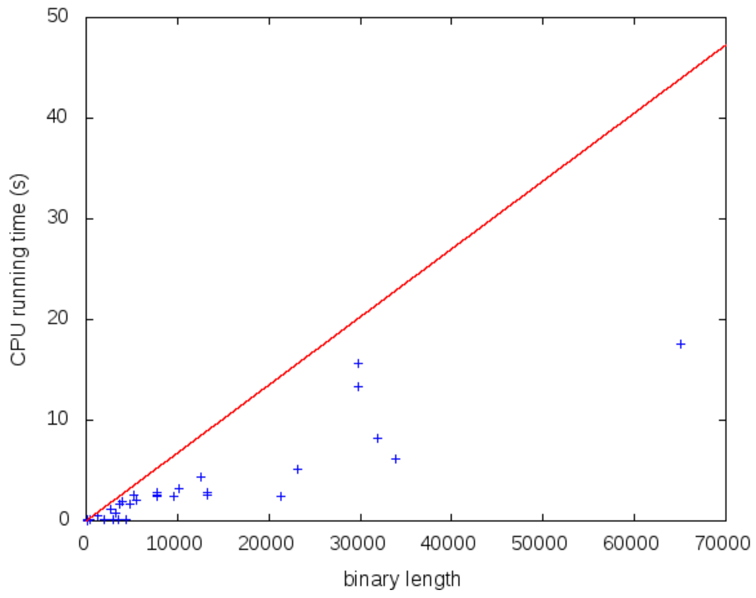

Figure 3: Plot with linear bound

Note: a lot of these testing examples turn out to have binary length $\langle A, Gr_A, l, u, b\rangle$ less that $10,000$. We tried very hard to find cases for which the binary length lies in the interval $(15,000; 70,000)$ and above but this seemed quite difficult. It is clear that for A of some size and larger, $\langle A, Gr_A, l, u, b\rangle$ is mostly dependent on $\langle Gr_A\rangle$ since it is then significantly larger than the other lengths. Since typically the (binary) length of the Graver basis for some matrix A is unpredictably large, we unfortunately were not able to get any control on the binary length while constructing examples. For example, in trying to find cases in the interval $(15,000; 70,000)$, we found several cases that involved too much memory so that the computer could not complete its computations.

5.3 Graver for a sum of squares

A widely studied sort of objective function for optimization problems is the function that can be written as a sum of squares. We consider these objective functions here since they are separable convex.
For the data we refer the reader to the .1- cases in Appendix B.2. Using these data we built up Figure 4, again involving 30 measure points.

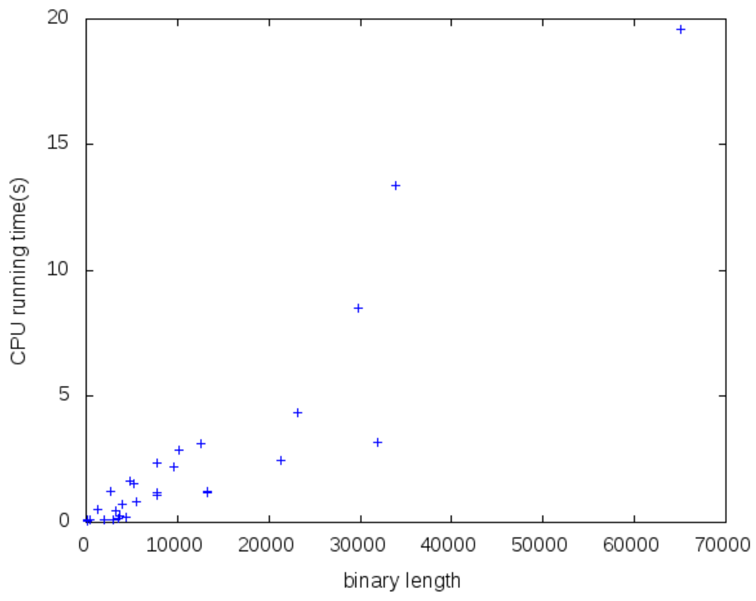

Figure 4: Plot for $f(x)$ a sum of squares

In the same way as for linear objective functions, we write $T(l) = cl^k$ and therefore $\log(T) = \log(c) + k\log(l)$ for $T(l), l \geq 0$. We plot the logarithm of the data in Figure 5, where we found that the points can be bounded by the line $\log(T) = -6.5 + 0.89\log(l)$, which is denoted by the red line. After again comparing both equalities, we find $c = e^{-6.5}$ and $k = 0.89$.
It is remarkable to see that there are no big differences in running time for linear functions and sums of squares. Intuitively, one could think that programming problems with 'simpler' functions like linear functions are solved faster than

problems with more 'difficult' separable convex functions. These data show this is not the case.

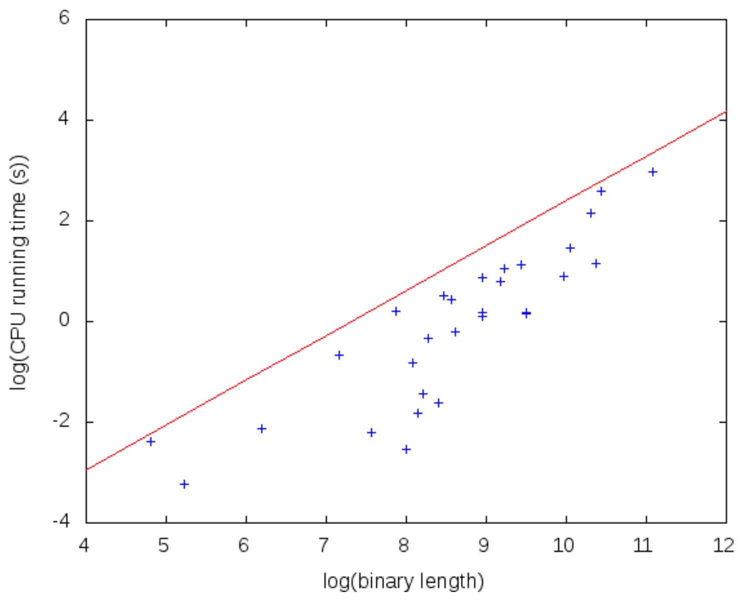

Figure 5: Log-plot for sum of squares

Studying the original data and the polynomial $T(l) = e^{-6.5}l^{0.89}$, we construct Figure 6 where the polynomial is the red line. It can be seen that the polynomial is indeed a bound for $T(l)$.

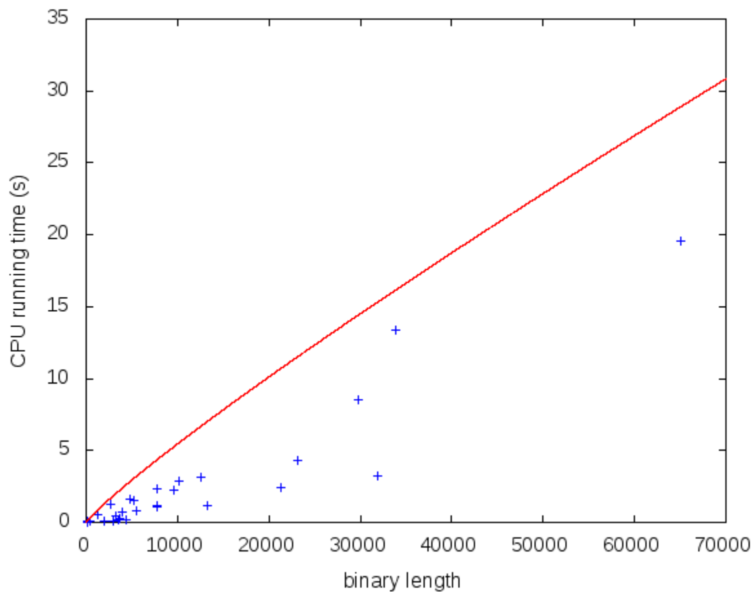

Figure 6: Polynomial bound

5.4 All Graver results together

When putting all points together in one plot, we have Figure 7.

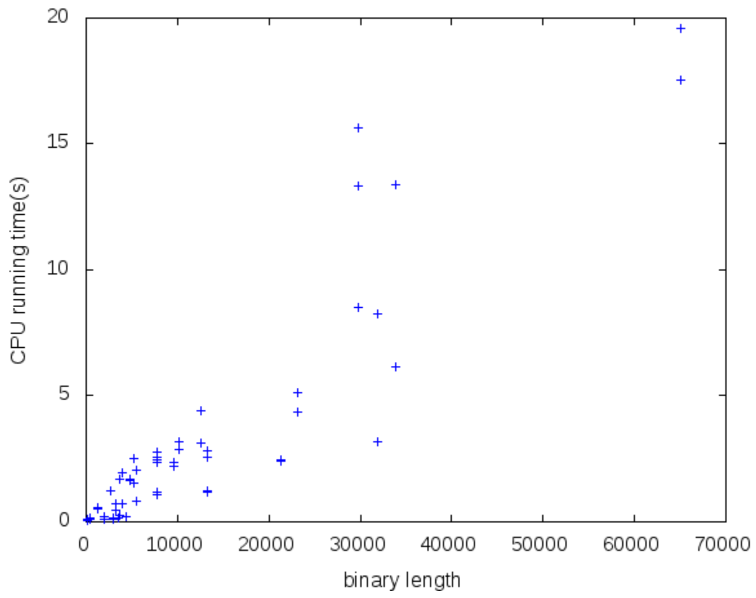

Figure 7: Plot of all 60 data points

Following the same procedure, we find that the data point can be bounded by the polynomial $T(l) = e^{-6.6}l^{0.93}$, again denoted by the red line in Figure 8.

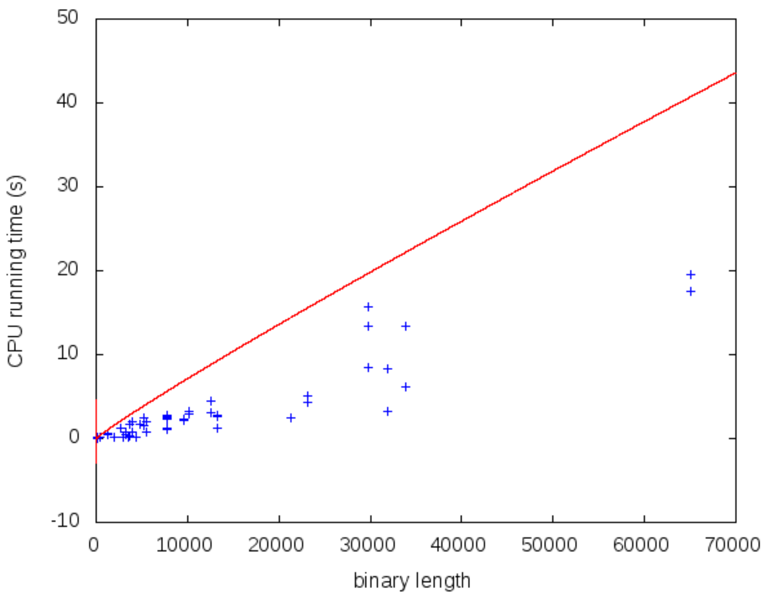

Figure 8: Bound on the running time

It must be noted that it would be desirable to have more points, especially for the 'empty' intervals. Then there could be stated a more general conclusion about the behavior of this algorithm as we have implemented it.

6 Conclusion

After some introduction on algebra and optimization, we started with theory on monomials and monomial ideals in the polynomial ring $k[x_1, ..., x_n]$. Next, Groebner bases were defined. We saw how they can be used to solve integer linear programming problems. The algorithm from [5] was implemented in Maple. Then Graver bases were introduced by the definition from Sturmfels. Also the definition from Onn was discussed, and was shown to be equivalent to the previous. We saw Sturmfels' algorithm for computing Graver bases and implemented it in Maple too. Onn gave an algorithm to solve an integer programming problem with a separable convex function by using Graver bases. We implemented it and compared the CPU runningtime with the time needed by the Groebner algorithm. The Graver algorithm seemed to be faster, although the computation of the Graver basis was not included. Another comparison was done on the Graver method studying integer programming problems with linear objective functions and with functions defined by sums of squares. Both types of functions are separable convex functions. It turned out that we can indeed find a polynomial bound on the running time, as shown by Onn. There were no big differences in running time between both kinds of functions.

7 Discussion

The implementation of the Groebner and Graver algorithm were done in Maple16. We experienced that Maple is not that suitable for solving problems over the integers. For example, the *isolve*-function, mentioned in section 4.6.1, can find integer solutions for a system of equalities, but not for a system of inequalities or a mixed system. Moreover, we think that problems may would been solved faster when using e.g. Matlab.
It is for this reason that we do not recommend Maple to solve integer programming problems. We would suggest to use Matlab for any further research on this topic. In particular, it would be very useful if one could develop a code package for Matlab, solving integer minimization and maximization problems.

References

[1] *S. Boyd, L. Vandenberghe*, `Convex Optimization`, Cambridge University Press, 2004.

[2] *C. Wendler*, `Groebner Bases with an Application to Integer Programming, Senior Exercise`, may 2004.

[3] *D.A. Cox, J. Little, D. O'Shea*, `Ideals, Varieties and Algorithms` , Springer-Verlag New York 1992.

[4] *D.A. Cox, J. Little, D. O'Shea*, `Using Algebraic Geometry`, 2nd ed., Springer 2004.

[5] *W.W. Adams, P. Loustaunau*, `An Introduction to Grobner Bases`, Graduate Studies in Mathematics, Volume 3, AMS 1994.

[6] *P. Conti, C. Traverso*, `Buchberger Algorithm and Integer Programming`, Istituto di Matematica Applicata, Universitá di Pisa, 1991.

[7] *B. Sturmfels*, `Grobner bases and convex polytopes`, University Lecture Series, Volume 8, American Mathematical Society, 1996.

[8] *S. Onn*, `Nonlinear Discrete Optimization, An Algorithmic Theory`, EMS, 2010.

[9] *T.W. Hungerford*, `Algebra`, Springer-Verlag 1974

[10] *J.A. De Loera, R. Hemmecke, M.Köppe*, `Algebraic and Geometric Ideas in the Theory of Discrete Optimization`, SIAM 2013.

[11] *D. Bertsimas, G. Perakis, S.Tayur*, `A New Algebraic Geometry Algorithm for Integer Programming`, Management Science 46.7, pp 999-1008, 2000.

[12] *R. Hemmecke, S. Onn, R. Weismantel*, `A polynomial oracle-time algorithm for convex integer minimization`, Mathematical Programming, Volume 126, Issue 1, pp 97-117, January 2011.

A Maple codes

A.1 Groebner Code

```
#This procedure computes a nonnegative integer solution
    x for the system Ax=b which minimizes the linear
    function f(x)=DotProduct(c1,x). A is an integer
    matrix of size m*n with nonnegative entries, all
    b[i], i=1...m, are nonnegative integers. Input c1 is
    given as a list.

Grobner:=proc(A,b,c1)
local st, m, n, X, Y, List, j, prd, i, K, Xlist, G, RHS,
    c, d, M, k, w, h, s, tijd;
uses LinearAlgebra, PolynomialIdeals, Groebner;

st:=time();
m:=RowDimension(A);
n:=ColumnDimension(A);
X := Vector(m, symbol = x);
Y := Vector(n, symbol = y);
List := [];

for j to n do
  prd := 1;
  for i to m do
    prd := prd*X[i]^A[i, j];
  end do;
  List := [op(List), Y[j]-prd];
end do;
K := PolynomialIdeal(List);
Xlist := [op(convert(Transpose(X),list)),
    op(convert(Transpose(Y),list))];

RHS:=1;
for i to m do
  RHS := RHS*X[i]^b[i];
end do;

c:=c1;
if min(c)<0 then
  d:=Vector(n);
  for j to n do
    d[j]:= add(A[i,j], i=1..m);
  end do;
  d:=convert(d,list);
  M:=[]; #This will be the list of minimal values of mu;
  for k to n do
    if c[k]<0 then
      M:=[op(M), -1*(c[k]/d[k])];
      end if;
    end do;
    c:=c+max(M)*d;
end if;

#Use w as weightvector for the x[i], they need to be
    bigger then the y[i].
w:=convert(Vector(m,max(c)+1),list);
G:=Groebner[Basis](K,wdeg([op(w),op(c)],Xlist));
h:=Reduce(RHS,G,wdeg([op(w),op(c)],Xlist));

#If h does not lie in k[y1,...,yn] then the system does
    not have a nonnegative integer solution.
for i to m do
  if not degree(h,x[i])=0 then
    error "No nonnegative integer solution x for Ax=b";
  end if;
end do;

s:=Vector(n);
for i to n do
  s[i]:=degree(h,y[i]);
end do;

tijd:=time()-st;
print(s,DotProduct(c1,s),tijd);
end proc;
```

A.2 Graver Code

A.2.1 Initial Point Algorithm

```
#This procedure finds an initial feasible solution x
    (integer vector of length n) of the set described by
    the linear equations from Ax=b (with A an integer
    matrix of size m*n) s.t. l<=x<=u. (l and u
    columnvectors of length n)

Initial:=proc(A,b,l,u)
local n, vect, X, eqs, v, i, s, V;
uses LinearAlgebra, Optimization;

n:=ColumnDimension(A);
vect:=Vector(n);
X:=Vector(n,symbol=x);
#First solve the system AX=b over the integers.
eqs:=convert(MatrixVectorMultiply(A,X)-b,set);
v:=isolve(eqs);
if v=NULL then
  error "No integer solution x for Ax=b";
end if;

for i to n do
  vect[i]:=rhs(v[i]);
end do;
#If there is just one solution found, we have to check
    whether it is bounded by l and u.
if n=Rank(A) then
  for i to n do
    if not (is(vect[i]>=l[i]) and is(vect[i]<=u[i])) then
      error "No integer solution x found between
          boundaries l and u (1)";
    end if;
  end do;
  return(vect);
  break;
end if;

V:={};
for i to n do
  V:={op(V), l[i]<=vect[i],vect[i]<=u[i]};
end do;
s:=LPSolve(1,V,assume=integer);
#if LPSolve doesn't find a feasible solution, it gives
    an error itself.
vect:=subs(s[2],vect);
end proc;
```

A.2.2 Graver basis Algorithm

```
#This procedure computes the Graver basis for a given
    matrix A of size m*n, output is a matrix where the
    rows are the basis elements (vectors in Z^n).
#We use the algorithm from Grobner Bases and Convex
    polytopes, B.Sturmfels, 1996, p.56.

Grbasis:=proc(A)
local n,r,X,Y,L,A1,Id,Alaw,Xlist, Gtor, k, H, vp, vm,
    K,p, i, pos, neg, v,G;
uses LinearAlgebra, Groebner;

n:=ColumnDimension(A);
r:=RowDimension(A);
X := Vector(n, symbol = x);
Y := Vector(n, symbol = y);
L:=[];
#Now start Sturmfels algorithm 7.2. Step 1: Choose term
    order on k[x,y] and compute reduced Groebner basis of
    I_law. First we construct Lawrence lifting of A.
A1:=<A|ZeroMatrix(r,n)>;
Id:=<IdentityMatrix(n,n)|IdentityMatrix(n,n)>;
Alaw:=<A1,Id>;
Xlist := [op(convert(Transpose(X),
    list)),op(convert(Transpose(Y),list))];
Gtor:=ToricIdealBasis(Alaw,Xlist,tdeg(op(Xlist)));

#Step 2: substitute all y_i =1 in Gtor. The resulting
    subset of k[x] is the Graver basis Grbasis of A.
for k to n do
  H:=subs(y[k]=1,Gtor);
  Gtor:=H;
end do;

vp:=Vector(n); #Automatically filled with zero's.
vm:=Vector(n);
K:=[];
#We want the Graver basis to be a matrix, where the rows
    are its elements. So from the x^vp-x^vm we have to
    find v=vp-vm.
for p in Gtor do
  if sign(op(1,p))=1 then
    pos:=op(1,p);
    neg:=-1*op(2,p);
  else
    pos:=op(2,p);
    neg:=-1*op(1,p);
  end if;

  for i to n do
    vp[i]:= degree(pos,x[i]);
    vm[i]:= degree(neg,x[i]);
  end do;
  v:=convert(vp-vm, list);
  K:=[op(K),v];
end do;
G:=convert(K,Matrix);
end proc;
```

A.2.3 Graver Algorithm

```
#This procedure computes an integer solution x for the
    system Ax=b, l<=x<=u; which minimizes the separable
    convex function f(x). A is an integer matrix of size
    m*n with nonnegative entries, all b[i], i=1...m, are
    nonnegative integers.
#Idea: Walk through all g_i's (p), select integer
    lambda's such that objective value of new point is
    smallest for fixed g_i. Then find minimal value,
    comparing the objective value of the p new points
    (x+lambda*g_i). If for best one yields
    f(x+lambda*g_i)<f(x), then replace x. Otherwise we
    are done.
# This is the main 'Graver-code'. It uses the procedures
    Grbasis and Initial.

Graver:=proc(A,b,f,l,u)
local st, G, x, p, q, again, Lambdas, i, List_u, List_l,
    g, j, lambda_min, lambda_max, df, r, v1, v2, y, K,
    v, w, z, tijd ;
uses LinearAlgebra, ListTools, RealDomain;

G:=Grbasis(A);

st:=time();
x:=Initial(A,b,l,u);
p:=RowDimension(G);
q:=ColumnDimension(G);
again:=true;
while again do
  Lambdas:=Vector(p);
  for i to p do
    List_u := [];
    List_l:=[];
    g:=Transpose(Row(G,i));
    for j to q do
      if g[j]>0 then
          List_u := [op(List_u),floor((u[j]-x[j])/g[j])];
          List_l := [op(List_l),ceil((l[j]-x[j])/g[j])];
      end if;
      if g[j]<0 then
          List_u := [op(List_u),floor((l[j]-x[j])/g[j])];
          List_l := [op(List_l),ceil((u[j]-x[j])/g[j])];
      end if;
    end do;

    lambda_min:=max(List_l);
    lambda_max:=min(List_u);
    if lambda_min=lambda_max then
      Lambdas[i]:=lambda_min;
      next;
    end if;

    #To find optimal integer lambda in interval [lambda_min,lambda_max] we set derivative d(f(t))/dt to zero.
    df:=diff(f(x+t*g),t);
    if df=0 then
      next;
    end if;

    r:=solve(df=0,t); #Since we use RealDomain it finds the only real solution. If f is a linear function then r=NULL
    if r=NULL then
      v1:=f(x+lambda_min*g);
      v2:=f(x+lambda_max*g);
      if v1<v2 then
        Lambdas[i]:=lambda_min;
      else
        Lambdas[i]:=lambda_max;
      end if;
    else  #We want lambda to be an integer. Check whether floor(r) or ceil(r) is best to take for lambda.
      y:= FindMinimalElement([subs(t=floor(r),
          f(x+t*g)), subs(t=ceil(r), f(x+t*g))],
          position);
      if y[2]=1 then
        Lambdas[i]:=floor(r);
      else
        Lambdas[i]:=ceil(r);
      end if;

      if not (lambda_min<=Lambdas[i] and
          Lambdas[i]<=lambda_max) then
        v1:=f(x+lambda_min*g);
        v2:=f(x+lambda_max*g);
```

```
          if v1<v2 then
            Lambdas[i]:=lambda_min;
          else
            Lambdas[i]:=lambda_max;
          end if;
        end if;
      end if;
    end do;

    #Construct Matrix K, where its rows are the points
        (vectors) x+lambda_i*g_{i}
    K:=[];
    for j to p do
      v:=convert(x+Lambdas[j]*Transpose(Row(G,j)),list);
      K:=[op(K),v];
    end do;
    convert(K,Matrix);

    #Construct list w of values w[i]=f(x+lambda_i*g_{i})
    w:=[];
    for j to p do
      w:=[op(w),f(Row(K,j))];
    end do;
    #Select minimal value of f(x+lambda_i*g_{i}). If this
        value is less that f(x), then replace x and repeat,
        else we are done.
    z:=FindMinimalElement(w,position); #Now z[1] is
        minimal value, z[2] is its position in w
    if z[1]<f(x) then
      x:=Transpose(Row(K,z[2]));
    else
      again:=false;
    end if;
  end do;

  tijd:=time()-st;
  print(x,f(x),tijd);
  end proc;
```

B Computed data

B.1 Grobner-Graver data

```
————————————————Graver data 1.2 – 10.2—————

"1.2,Graver(Matrix([[4,2,6,0],[2,4,0,6],[4,2,2,4]]),
    Vector([6,6,6]), x-> x[3], Vector(4,0), Vector(4,100));"
```

```
[1]
[ ]
[1]
[ ], 0, 0.071
[0]
[ ]
[0]
```

```
"2.2,Graver(Matrix([[4,2,6,0,2],[1,2,6,0,5],[1,3,2,2,1]]),
    Vector([8,8,5]), x-> x[5] ,Vector(5,0), Vector(5,100));"
```

```
[0]
[ ]
[1]
[ ]
[1], 0, 0.020
[ ]
[0]
[ ]
[0]
```

```
"3.2,Graver(Matrix([[3,1,2,1,2,1,0],[3,0,1,2,1,1,2],[1,2,1,1,2,0,1]]),
    Vector([4,4,3]), x-> x[3]+x[6], Vector(7,0),
    Vector(7,100));"
```

```
[0]
[ ]
[0]
[ ]
[0]
[ ]
[1], 1, 0.219
[ ]
[1]
[ ]
[1]
[ ]
[0]
```

```
"4.2,Graver(Matrix([[2,0,1,2,1,0,2],[1,2,0,1,2,1,0],[0,1,1,0,1,2,2]]),
    Vector([2,1,0]), x-> x[1]+x[5], Vector(7,0),
    Vector(7,100));"
```

```
[0]
[ ]
```

```
                              [0]
                              [ ]
                              [0]
                              [ ]
                              [1], 0, 0.080
                              [ ]
                              [0]
                              [ ]
                              [0]
                              [ ]
                              [0]
```

```
"5a.2,Graver(Matrix([[2,0,1,2,1,0,2],[1,2,0,1,2,1,0],
    [1,0,0,1,1,0,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    x-> x[1]+x[5]  , Vector(7,0), Vector(7,100));"
```

```
                              [0]
                              [ ]
                              [0]
                              [ ]
                              [0]
                              [ ]
                              [1], 0, 0.030
                              [ ]
                              [0]
                              [ ]
                              [0]
                              [ ]
                              [0]
```

```
"5b.2,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,0,1,2,1,0],
    [1,0,0,1,1,0,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    x-> x[1]+x[5]  , Vector(7,0), Vector(7,100));"
```

```
                              [0]
                              [ ]
                              [0]
                              [ ]
                              [0]
                              [ ]
                              [1], 0, 0.100
                              [ ]
                              [0]
                              [ ]
                              [0]
                              [ ]
                              [0]
```

```
"5c.2,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
    [1,0,0,1,1,0,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    x-> x[1]+x[5]  , Vector(7,0), Vector(7,100));"
```

```
                              [0]
```

```
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [1],  0,  0.170
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
```

```
"5d.2,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
   [1,3,1,1,4,2,1],  [0,1,1,0,1,2,2]]),  Vector([2,1,1,0]),
   x-> x[1]+x[5],  Vector(7,0),  Vector(7,100));"
```

```
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [1],  0,  2.180
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
```

```
"5e.2,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
   [1,3,1,1,4,2,1],  [1,1,2,0,1,2,2]]),  Vector([2,1,1,0]),
   x-> x[1]+x[5],  Vector(7,0),  Vector(7,100));  "
```

```
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [1],  0,  0.120
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
```

```
"6.2,Graver(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],
   [1,3,3,1,4,2,1],  [1,1,2,7,1,0,5]]),  Vector([4,5,6,6]),
   x-> x[1]+x[5]-x[6]   ,  Vector(7,0),  Vector(7,100));"
```

```
[0]
[ ]
[1]
[ ]
[0]
[ ]
[0], −1, 0.351
[ ]
[0]
[ ]
[1]
[ ]
[1]
```

```
"7.2,Graver(Matrix([[4,1,5,2,0,1,2],[3,2,1,3,6,1,0],
    [1,1,2,1,1,0,0], [6,2,3,5,0,2,1]]), Vector([4,3,1,6]),x->
    2*x[1]-x[5], Vector(7,0), Vector(7,100));"
```

```
[0]
[ ]
[0]
[ ]
[0]
[ ]
[1], 0, 0.570
[ ]
[0]
[ ]
[0]
[ ]
[1]
```

```
"8.2,Graver(Matrix([[5,2,1,3,0,1,2],[1,3,2,5,4,0,1],
    [1,1,2,3,1,0,5],[1,2,3,5,0,2,1]]), Vector([3,5,3,5]),x->
    x[4], Vector(7,0), Vector(7,100));"
```

```
[0]
[ ]
[1]
[ ]
[1]
[ ]
[0], 0, 5.541
[ ]
[0]
[ ]
[0]
[ ]
[0]
```

```
"9.2,Graver(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],
    [1,3,3,1,4,2,1], [1,1,2,7,1,0,5], [1,2,3,2,0,2,1]]),
    Vector([4,5,6,6,5]), x-> x[3]+x[5] , Vector(7,0),
    Vector(7,100));"
```

```
                                        [0]
                                        [ ]
                                        [1]
                                        [ ]
                                        [0]
                                        [ ]
                                        [0],  0,  0.049
                                        [ ]
                                        [0]
                                        [ ]
                                        [1]
                                        [ ]
                                        [1]
```

```
"10.2,Graver(Matrix([[7,2,1,3,6,1,2],[1,3,3,1,4,2,1],
    [1,1,2,7,1,0,5],  [1,2,3,2,0,2,1]]),  Vector([5,6,6,5]),
    x-> 3*x[4]   ,  Vector(7,0),  Vector(7,100));   "
```

```
                                        [0]
                                        [ ]
                                        [1]
                                        [ ]
                                        [0]
                                        [ ]
                                        [0],  0,  0.391
                                        [ ]
                                        [0]
                                        [ ]
                                        [1]
                                        [ ]
                                        [1]
```

```
——————————————————Grobner  data  1.2  –  10.2—————
"1.2,Grobner(Matrix([[4,2,6,0],[2,4,0,6],[4,2,2,4]]),Vector([6,6,6]),
    [0,0,1,0]);"
```

```
                                        [1]
                                        [ ]
                                        [1]
                                        [ ],  0,  0.080
                                        [0]
                                        [ ]
                                        [0]
```

```
"2.2,Grobner(Matrix([[4,2,6,0,2],[1,2,6,0,5],[1,3,2,2,1]]),
    Vector([8,8,5]),  [0,0,0,0,1]);"
```

```
                                        [0]
                                        [ ]
                                        [1]
                                        [ ]
                                        [1],  0,  0.140
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]

"3.2,Grobner(Matrix([[3,1,2,1,2,1,0],[3,0,1,2,1,1,2],
    [1,2,1,1,2,0,1]]), Vector([4,4,3]), [0,0,1,0,0,1,0]);"

                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [1],  1,  2.399
                                        [ ]
                                        [1]
                                        [ ]
                                        [1]
                                        [ ]
                                        [0]

"4.2,Grobner(Matrix([[2,0,1,2,1,0,2], [1,2,0,1,2,1,0],
    [0,1,1,0,1,2,2]]), Vector([2,1,0]), [1,0,0,0,1,0,0]);"

                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [1],  0,  0.401
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]

"5a.2,Grobner(Matrix([[2,0,1,2,1,0,2],[1,2,0,1,2,1,0],
    [1,0,0,1,1,0,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    [1,0,0,0,1,0,0]);"

                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [0]
                                        [ ]
                                        [1],  0,  0.610
                                        [ ]
                                        [0]
                                        [ ]
```

```
[0]
[ ]
[0]
```

```
"5b.2,Grobner(Matrix([[2,1,5,2,1,1,2],[1,2,0,1,2,1,0],
    [1,0,0,1,1,0,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    [1,0,0,0,1,0,0]);"
```

```
[0]
[ ]
[0]
[ ]
[0]
[ ]
[1], 0, 2105.650
[ ]
[0]
[ ]
[0]
[ ]
[0]
```

```
"5c.2,Grobner(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
    [1,0,0,1,1,0,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    [1,0,0,0,1,0,0]);"
```

```
[0]
[ ]
[0]
[ ]
[0]
[ ]
[1], 0, 31.080
[ ]
[0]
[ ]
[0]
[ ]
[0]
```

```
"5d.2,Grobner(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
    [1,3,1,1,4,2,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    [1,0,0,0,1,0,0]);"
```

```
[0]
[ ]
[0]
[ ]
[0]
[ ]
[1], 0, 188.320
[ ]
[0]
[ ]
```

```
                              [0]
                              [ ]
                              [0]

"5e.2,Grobner(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
    [1,3,1,1,4,2,1], [1,1,2,0,1,2,2]]), Vector([2,1,1,0]),
    [1,0,0,0,1,0,0]);"

                               [0]
                               [ ]
                               [0]
                               [ ]
                               [0]
                               [ ]
                               [1], 0, 92.450
                               [ ]
                               [0]
                               [ ]
                               [0]
                               [ ]
                               [0]

"6.2,Grobner(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],
    [1,3,3,1,4,2,1], [1,1,2,7,1,0,5]]), Vector([4,5,6,6]),
    [1,0,0,0,1,-1,0]);"

Error, (in Groebner:-F4:-GroebnerBasis) Maple was unable to
    allocate enough memory to complete this computation.
    Please see ?alloc

"7.2,Grobner(Matrix([[4,1,5,2,0,1,2],[3,2,1,3,6,1,0],
    [1,1,2,1,1,0,0], [6,2,3,5,0,2,1]]), Vector([4,3,1,6]),
    [2,0,0,0,-1,0,0]);"

                               [0]
                               [ ]
                               [0]
                               [ ]
                               [0]
                               [ ]
                               [1], 0, 5.221
                               [ ]
                               [0]
                               [ ]
                               [0]
                               [ ]
                               [1]

"8.2,Grobner(Matrix([[5,2,1,3,0,1,2],[1,3,2,5,4,0,1],
    [1,1,2,3,1,0,5], [1,2,3,5,0,2,1]]),
    Vector([3,5,3,5]),[0,0,0,1,0,0,0]);"

Error, (in divmon) Maple was unable to allocate enough memory
    to complete this computation. Please see ?alloc
```

```
"9.2,Grobner(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],
    [1,3,3,1,4,2,1], [1,1,2,7,1,0,5], [1,2,3,2,0,2,1]]),
    Vector([4,5,6,6,5]), [0,0,1,0,1,0,0]);"

Error, (in Groebner:-F4:-GroebnerBasis) Maple was unable to
    allocate enough memory to complete this computation.
    Please see ?alloc

"10.2,Grobner(Matrix([[7,2,1,3,6,1,2],[1,3,3,1,4,2,1],
    [1,1,2,7,1,0,5], [1,2,3,2,0,2,1]]), Vector([5,6,6,5]),
    [0,0,0,3,0,0,0]);"

Error, (in Groebner:-F4:-GroebnerBasis) Maple was unable to
    allocate enough memory to complete this computation.
    Please see ?alloc
```

B.2 Graver data

```
"1.1,Graver(Matrix([[4,2,6,0],[2,4,0,6],[4,2,2,4]]),Vector([6,6,6]),
    x-> x[3]^2+x[1]^2 , Vector(4,-100), Vector(4,100));"

                              [1]
                              [ ]
                              [1]
                              [ ],  1,  0.091
                              [0]
                              [ ]
                              [0]

"2.1,Graver(Matrix([[4,2,6,0,2],[1,2,6,0,5],[1,3,2,2,1]]),
    Vector([8,8,5]), x-> x[3]^2+x[4]^2 ,Vector(5,-100),
    Vector(5,100));"

                              [1]
                              [ ]
                              [1]
                              [ ]
                              [0],  0,  0.040
                              [ ]
                              [0]
                              [ ]
                              [1]


"1.2,Graver(Matrix([[4,2,6,0],[2,4,0,6],[4,2,2,4]]),
    Vector([6,6,6]), x-> x[3], Vector(4,-100),
    Vector(4,100));"

                            [100]
                            [   ]
                            [100]
```

```
                                        [    ],  -99,  0.070
                                        [-99]
                                        [    ]
                                        [-99]
```

```
"2.2,Graver(Matrix([[4,2,6,0,2],[1,2,6,0,5],[1,3,2,2,1]]),
    Vector([8,8,5]),  x-> x[5]   ,Vector(5,-100),
    Vector(5,100));"
```

```
                                       [-100]
                                       [     ]
                                       [    7]
                                       [     ]
                                       [   99],  -100,  0.040
                                       [     ]
                                       [   -7]
                                       [     ]
                                       [-100]
```

```
"3.1,Graver(Matrix([[3,1,2,1,2,1,0],[3,0,1,2,1,1,2],
    [1,2,1,1,2,0,1]]),  Vector([4,4,3]),  x-> x[3]^2+x[6]^2   ,
    Vector(7,-100),  Vector(7,100));"
```

```
                                          [  3]
                                          [   ]
                                          [  5]
                                          [   ]
                                          [  0]
                                          [   ]
                                          [  0],  0,  1.221
                                          [   ]
                                          [-5]
                                          [   ]
                                          [  0]
                                          [   ]
                                          [  0]
```

```
"3.2,Graver(Matrix([[3,1,2,1,2,1,0],[3,0,1,2,1,1,2],
    [1,2,1,1,2,0,1]]),  Vector([4,4,3]),  x-> x[3]+x[6],
    Vector(7,-100),  Vector(7,100));"
```

```
                                       [   98]
                                       [     ]
                                       [   85]
                                       [     ]
                                       [-100]
                                       [     ]
                                       [   85],  -200,  1.210
                                       [     ]
                                       [  -80]
                                       [     ]
                                       [-100]
                                       [     ]
                                       [  -90]
```

```
"4.1,Graver(Matrix([[2,0,1,2,1,0,2],[1,2,0,1,2,1,0],
    [0,1,1,0,1,2,2]]), Vector([2,1,0]), x-> x[1]^2+x[5]^2 ,
    Vector(7,-100), Vector(7,100));"
```

```
[0]
[ ]
[0]
[ ]
[0]
[ ]
[1],  0,  0.510
[ ]
[0]
[ ]
[0]
[ ]
[0]
```

```
"4.2,Graver(Matrix([[2,0,1,2,1,0,2],[1,2,0,1,2,1,0],
    [0,1,1,0,1,2,2]]), Vector([2,1,0]), x-> x[1]+x[5],
    Vector(7,-100), Vector(7,100));"
```

```
[-99]
[   ]
[100]
[   ]
[ 25]
[   ]
[100],  -174,  0.540
[   ]
[-75]
[   ]
[-50]
[   ]
[ 25]
```

```
"5a.1,Graver(Matrix([[2,0,1,2,1,0,2], [1,2,0,1,2,1,0],
    [1,0,0,1,1,0,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    x-> x[1]^2+x[5]^2 , Vector(7,-100), Vector(7,100));"
```

```
[ 0]
[  ]
[-4]
[  ]
[ 0]
[  ]
[ 4],  0,  0.120
[  ]
[ 0]
[  ]
[ 5]
[  ]
[-3]
```

```
"5a.2,Graver(Matrix([[2,0,1,2,1,0,2],[1,2,0,1,2,1,0],[1,0,0,1,1,0,1],
    [0,1,1,0,1,2,2]]), Vector([2,1,1,0]), x-> x[1]+x[5]  ,
    Vector(7,-100), Vector(7,100));"
```

```
[  -74]
[     ]
[  100]
[     ]
[-100]
[     ]
[  100], -174, 0.151
[     ]
[-100]
[     ]
[  -25]
[     ]
[   75]
```

```
"5b.1,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,0,1,2,1,0],
    [1,0,0,1,1,0,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    x-> x[1]^2+x[5]^2  , Vector(7,-100), Vector(7,100));"
```

```
[0]
[ ]
[0]
[ ]
[0]
[ ]
[1], 0, 0.439
[ ]
[0]
[ ]
[0]
[ ]
[0]
```

```
"5b.2,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,0,1,2,1,0],
    [1,0,0,1,1,0,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    x-> x[1]+x[5]  , Vector(7,-100), Vector(7,100));"
```

```
[-99]
[    ]
[ 99]
[    ]
[-18]
[    ]
[100], -162, 0.730
[    ]
[-63]
[    ]
[-72]
[    ]
[ 63]
```

```
"5c.1,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
    [1,0,0,1,1,0,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    x-> x[1]^2+x[5]^2 , Vector(7,-100), Vector(7,100));"
```

```
                                [0]
                                [ ]
                                [0]
                                [ ]
                                [0]
                                [ ]
                                [1], 0, 0.809
                                [ ]
                                [0]
                                [ ]
                                [0]
                                [ ]
                                [0]
```

```
"5c.2,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
    [1,0,0,1,1,0,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    x-> x[1]+x[5] , Vector(7,-100), Vector(7,100));"
```

```
                             [-100]
                             [    ]
                             [ 100]
                             [    ]
                             [ -12]
                             [    ]
                             [  89], -132, 2.030
                             [    ]
                             [ -32]
                             [    ]
                             [ -72]
                             [    ]
                             [  44]
```

```
"5d.1,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
    [1,3,1,1,4,2,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    x-> x[1]^2+x[5]^2 , Vector(7,-100), Vector(7,100));"
```

```
                                [0]
                                [ ]
                                [0]
                                [ ]
                                [0]
                                [ ]
                                [1], 0, 8.530
                                [ ]
                                [0]
                                [ ]
                                [0]
                                [ ]
                                [0]
```

```
"5d.2,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
   [1,3,1,1,4,2,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
   x-> x[1]+x[5], Vector(7,-100), Vector(7,100));"
```

```
                              [-100]
                              [    ]
                              [ 100]
                              [    ]
                              [ -13]
                              [    ]
                              [  78], -135, 15.660
                              [    ]
                              [ -35]
                              [    ]
                              [ -98]
                              [    ]
                              [  72]
```

```
"5e.1,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
   [1,3,1,1,4,2,1], [1,1,2,0,1,2,2]]), Vector([2,1,1,0]),
   x-> x[1]^2+x[5]^2  , Vector(7,-100), Vecto(7,100)); "
```

```
                                 [0]
                                 [ ]
                                 [0]
                                 [ ]
                                 [0]
                                 [ ]
                                 [1], 0, 0.720
                                 [ ]
                                 [0]
                                 [ ]
                                 [0]
                                 [ ]
                                 [0]
```

```
"5e.2,Graver(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
   [1,3,1,1,4,2,1], [1,1,2,0,1,2,2]]), Vector([2,1,1,0]),
   x-> x[1]+x[5], Vector(7,-100), Vector(7,100)); "
```

```
                              [-100]
                              [    ]
                              [  99]
                              [    ]
                              [  -3]
                              [    ]
                              [  17], -139, 1.930
                              [    ]
                              [ -39]
                              [    ]
                              [ -77]
```

```
                                    [       ]
                                    [  100]

"6.1,Graver(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],
    [1,3,3,1,4,2,1], [1,1,2,7,1,0,5]]), Vector([4,5,6,6]),
    x-> x[1]^2+x[5]^2  , Vector(7,-100), Vector(7,100));"

                                     [0]
                                     [ ]
                                     [1]
                                     [ ]
                                     [0]
                                     [ ]
                                     [0],  0,  1.140
                                     [ ]
                                     [0]
                                     [ ]
                                     [1]
                                     [ ]
                                     [1]

"6.2,Graver(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],
    [1,3,3,1,4,2,1], [1,1,2,7,1,0,5]]), Vector([4,5,6,6]),
    x-> x[1]+x[5]-x[6]  , Vector(7,-100), Vector(7,100));"

                                   [  72]
                                   [    ]
                                   [-49]
                                   [    ]
                                   [  97]
                                   [    ]
                                   [  36],  -121,  2.830
                                   [    ]
                                   [-93]
                                   [    ]
                                   [100]
                                   [    ]
                                   [-74]

"7.1,Graver(Matrix([[4,1,5,2,0,1,2],[3,2,1,3,6,1,0],
    [1,1,2,1,1,0,0], [6,2,3,5,0,2,1]]), Vector([4,3,1,6]),x->
    2*x[1]^2+x[5]^2, Vector(7,-100), Vector(7,100));"

                                     [0]
                                     [ ]
                                     [0]
                                     [ ]
                                     [0]
                                     [ ]
                                     [1],  0,  2.870
                                     [ ]
                                     [0]
                                     [ ]
                                     [0]
```

```
                              [ ]
                              [1]

"7.2,Graver(Matrix([[4,1,5,2,0,1,2],[3,2,1,3,6,1,0],
    [1,1,2,1,1,0,0], [6,2,3,5,0,2,1]]), Vector([4,3,1,6]),x->
    2*x[1]-x[5], Vector(7,-100), Vector(7,100));"

                           [-100]
                           [    ]
                           [ -85]
                           [    ]
                           [  40]
                           [    ]
                           [  96], -210, 3.200
                           [    ]
                           [  10]
                           [    ]
                           [  85]
                           [    ]
                           [   6]

"8.1,Graver(Matrix([[5,2,1,3,0,1,2],[1,3,2,5,4,0,1],
    [1,1,2,3,1,0,5], [1,2,3,5,0,2,1]]), Vector([3,5,3,5]),x->
    x[1]^2+x[5]^2, Vector(7,-100), Vector(7,100));"

                              [0]
                              [ ]
                              [0]
                              [ ]
                              [0]
                              [ ]
                              [1], 0, 19.580
                              [ ]
                              [0]
                              [ ]
                              [0]
                              [ ]
                              [0]

"8.2,Graver(Matrix([[5,2,1,3,0,1,2],[1,3,2,5,4,0,1],
    [1,1,2,3,1,0,5], [1,2,3,5,0,2,1]]), Vector([3,5,3,5]),x->
    x[4], Vector(7,-100), Vector(7,100));"

                           [  19]
                           [    ]
                           [  12]
                           [    ]
                           [ 100]
                           [    ]
                           [-100], -100, 20.880
                           [    ]
                           [  62]
                           [    ]
                           [  80]
[       ]
[      2]
```

```
"9.1,Graver(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],
     [1,3,3,1,4,2,1], [1,1,2,7,1,0,5], [1,2,3,2,0,2,1]]),
     Vector([4,5,6,6,5]), x-> x[1]^2+x[5]^2 , Vector(7,-100),
     Vector(7,100));"
```

```
[0]
[ ]
[1]
[ ]
[0]
[ ]
[0], 0, 0.080
[ ]
[0]
[ ]
[1]
[ ]
[1]
```

```
"9.2,Graver(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],[1,3,3,1,4,2,1],
     [1,1,2,7,1,0,5], [1,2,3,2,0,2,1]]), Vector([4,5,6,6,5]),
     x-> x[3]+x[5] , Vector(7,-100), Vector(7,100));"
```

```
[ -4]
[    ]
[-44]
[    ]
[-23]
[    ]
[-25], -18, 0.150
[    ]
[  5]
[    ]
[ 81]
[    ]
[ 54]
```

```
"10.1,Graver(Matrix([[7,2,1,3,6,1,2],[1,3,3,1,4,2,1],
     [1,1,2,7,1,0,5], [1,2,3,2,0,2,1]]), Vector([5,6,6,5]),
     x-> x[1]^2+2*x[5]^2 , Vector(7,-100), Vector(7,100)); "
```

```
[   0]
[    ]
[-21]
[    ]
[-12]
[    ]
[-22], 0, 3.160
[    ]
[   0]
[    ]
```

```
                    [ 43]
                    [    ]
                    [ 41]

"10.2,Graver(Matrix([[7,2,1,3,6,1,2],[1,3,3,1,4,2,1],
    [1,1,2,7,1,0,5], [1,2,3,2,0,2,1]]), Vector([5,6,6,5]),
    x-> 3*x[4] , Vector(7,-100), Vector(7,100)); "

                    [ 36]
                    [    ]
                    [ -8]
                    [    ]
                    [ 92]
                    [    ]
                    [-97], -291, 4.420
                    [    ]
                    [-22]
                    [    ]
                    [-98]
                    [    ]
                    [ 99]

"11.1,Graver(Matrix([[1,0,1,4,2,6,0],[2,4,2,1,0,0,1],[4,2,3,0,2,2,4],
    [0,1,2,3,2,0,1], [1,1,0,2,1,2,1]]), Vector([1,7,10,2,3]),
    x-> x[3]^2 , Vector(7,-100), Vector(7,100));"

                    [1]
                    [ ]
                    [1]
                    [ ]
                    [0]
                    [ ]
                    [0], 0, 0.109
                    [ ]
                    [0]
                    [ ]
                    [0]
                    [ ]
                    [1]

"11.2,Graver(Matrix([[1,0,1,4,2,6,0],[2,4,2,1,0,0,1],[4,2,3,0,2,2,4],
    [0,1,2,3,2,0,1], [1,1,0,2,1,2,1]]), Vector([1,7,10,2,3]),
    x-> 4*x[3] , Vector(7,-100), Vector(7,100));"

                    [ 41]
                    [    ]
                    [ 21]
                    [    ]
                    [-30]
                    [    ]
                    [-30], -120, 0.181
                    [    ]
                    [100]
                    [    ]
```

```
[-15]
[   ]
[-69]
```

```
"12.1,Graver(Matrix([[1,1,0,2,1,2,0,2,1],[3,2,1,0,1,1,2,0,2],
    [1,0,1,1,2,1,1,0,1], [1,2,1,0,1,0,1,2,2],
    [1,2,0,1,2,0,1,2,2]]), Vector([5,4,4,2,3]), x->
    x[5]^2+x[8]^2  , Vector(9,-100), Vector(9,100)); "
```

```
[ 2]
[  ]
[-2]
[  ]
[ 0]
[  ]
[ 1]
[  ]
[ 0], 0, 2.370
[  ]
[ 0]
[  ]
[-2]
[  ]
[ 0]
[  ]
[ 3]
```

```
"12.2,Graver(Matrix([[1,1,0,2,1,2,0,2,1],[3,2,1,0,1,1,2,0,2],
    [1,0,1,1,2,1,1,0,1], [1,2,1,0,1,0,1,2,2],
    [1,2,0,1,2,0,1,2,2]]), Vector([5,4,4,2,3]), x->x[5]  ,
    Vector(9,-100), Vector(9,100)); "
```

```
[  10]
[    ]
[ -98]
[    ]
[  -8]
[    ]
[  93]
[    ]
[-100], -100, 2.570
[    ]
[ -60]
[    ]
[  92]
[    ]
[  25]
[    ]
[  77]
```

```
"13.1,
    Graver(Matrix([[1,1,0,2,1,2,0,2,1],[3,2,1,0,1,1,2,0,2],
    [1,0,1,1,2,1,1,0,1], [1,2,1,0,1,0,1,2,2],
    [1,2,0,1,2,0,1,2,2]]), Vector([5,4,4,2,3]), x-> x[5]^2  ,
    Vector(9,-100), Vector(9,100)); "

                              [ 2]
                              [  ]
                              [-2]
                              [  ]
                              [ 0]
                              [  ]
                              [ 1]
                              [  ]
                              [ 0], 0, 1.100
                              [  ]
                              [ 0]
                              [  ]
                              [ 0]
                              [  ]
                              [ 1]
                              [  ]
                              [ 1]

"13.2,
    Graver(Matrix([[1,1,0,2,1,2,0,2,1],[3,2,1,0,1,1,2,0,2],
    [1,0,1,1,2,1,1,0,1], [1,2,1,0,1,0,1,2,2],
    [1,2,0,1,2,0,1,2,2]]), Vector([5,4,4,2,3]), x->
    x[5]+x[7], Vector(9,-100), Vector(9,100)); "

                             [ 68]
                             [   ]
                             [-46]
                             [   ]
                             [ 44]
                             [   ]
                             [100]
                             [   ]
                             [-55], -154, 2.450
                             [   ]
                             [-99]
                             [   ]
                             [-99]
                             [   ]
                             [-32]
                             [   ]
                             [100]

"14.1,
    Graver(Matrix([[1,1,0,2,1,2,0,2,1],[3,2,1,0,1,1,2,0,2],
    [1,0,1,1,2,1,1,0,1], [1,2,1,0,1,0,1,2],
    [1,2,0,1,2,0,1,2]]), Vector([5,4,4,2,3]), x->
    x[5]^2+x[7]^2, Vector(9,-100), Vector(9,100)); "

                              [ 2]
                              [  ]
                              [-2]
                              [  ]
                              [  0]
                              [   ]
                              [  1]
                              [   ]
                              [  0],  0,  1.190
                              [   ]
                              [  0]
                              [   ]
                              [  0]
                              [   ]
                              [  1]
                              [   ]
                              [  1]

"14.2,
    Graver(Matrix([[1,1,0,2,1,2,0,2,1],[3,2,1,0,1,1,2,0,2],
    [1,0,1,1,2,1,1,0,1],  [1,2,1,0,1,0,1,2],
    [1,2,0,1,2,0,1,2]]),  Vector([5,4,4,2,3]),  x->
    2*x[5]-x[7],  Vector(9,-100),  Vector(9,100)); "
                           [  -18]
                           [     ]
                           [  -78]
                           [     ]
                           [  -10]
                           [     ]
                           [   91]
                           [     ]
                           [-100],  -300,  2.790
                           [     ]
                           [  -42]
                           [     ]
                           [  100]
                           [     ]
                           [   10]
                           [     ]
                           [   83]

"15.1,
    Graver(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],[1,3,3,1,4,2,1],
    [1,1,2,7,1,0,5]]),  Vector([4,5,6,6]),  x-> x[1]^2+x[5]^2,
    Vector(7,-100),  Vector(7,100)); "

                              [0]
                              [ ]
                              [1]
                              [ ]
                              [0]
                              [ ]
                              [0],  0,  1.170
                              [ ]
                              [0]
                              [ ]
                              [1]
                              [ ]
```

```
                         [1]


"15.2, Graver(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],
    [1,3,3,1,4,2,1], [1,1,2,7,1,0,5]]), Vector([4,5,6,6]),
    x-> x[1]+x[5]-x[6], Vector(7,-100), Vector(7,100)); "

                       [ 72]
                       [   ]
                       [-49]
                       [   ]
                       [ 97]
                       [   ]
                       [ 36], -121, 2.570
                       [   ]
                       [-93]
                       [   ]
                       [100]
                       [   ]
                       [-74]


"16.1, Graver(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
    [1,3,1,1,4,2,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    x-> x[1]^2+x[5]^2 , Vector(7,-100), Vector(7,100)); "

                         [0]
                         [ ]
                         [0]
                         [ ]
                         [0]
                         [ ]
                         [1], 0, 8.530
                         [ ]
                         [0]
                         [ ]
                         [0]
                         [ ]
                         [0]

"16.2, Graver(Matrix([[2,1,5,2,1,1,2],[1,2,1,1,6,1,2],
    [1,3,1,1,4,2,1], [0,1,1,0,1,2,2]]), Vector([2,1,1,0]),
    x-> x[1]-4*x[5] , Vector(7,-100), Vector(7,100)); "

                      [-100]
                      [    ]
                      [ -44]
                      [    ]
                      [  96]
                      [    ]
                      [ -67], -308, 11.970
                      [    ]
                      [  52]
                      [    ]
```

```
                                        [   48]
                                        [     ]
                                        [-100]
"17.1,Graver(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],[1,6,3,1,4,2,7],
    [3,1,2,6,7,1,9], [1,2,3,5,6,2,7]]), Vector([5,5,7,6,5]),
    x-> x[1]^2+x[5]^2  , Vector(7,-100), Vector(7,100));"

                                          [  2]
                                          [   ]
                                          [ -9]
                                          [   ]
                                          [ -5]
                                          [   ]
                                          [-11], 13, 0.241
                                          [   ]
                                          [  3]
                                          [   ]
                                          [ 19]
                                          [   ]
                                          [  5]

"17.2,Graver(Matrix([[4,1,5,2,8,1,2],[7,2,1,3,6,1,2],[1,6,3,1,4,2,7],
    [3,1,2,6,7,1,9], [1,2,3,5,6,2,7]]), Vector([5,5,7,6,5]),
    x-> x[1]+x[6]  , Vector(7,-100), Vector(7,100));"

                                         [-58]
                                         [   ]
                                         [ 47]
                                         [   ]
                                         [-72]
                                         [   ]
                                         [  5], -122, 0.171
                                         [   ]
                                         [ 83]
                                         [   ]
                                         [-64]
                                         [   ]
                                         [-30]

"18.1,Graver(Matrix([[7,2,1,3,6,1,2],[1,3,3,1,4,2,1],[1,6,2,7,1,0,5],
    [1,2,4,2,8,2,1]]), Vector([5,6,6,5]), x-> x[1]^2+2*x[5]^2
    , Vector(7,-100), Vector(7,100)); "

                                           [ 0]
                                           [  ]
                                           [ 1]
                                           [  ]
                                           [-1]
                                           [  ]
                                           [ 1], 0, 4.340
                                           [  ]
                                           [ 0]
                                           [  ]
                                           [ 3]
```

```
                                    [    ]
                                    [-1]

"18.2,Graver(Matrix([[7,2,1,3,6,1,2],[1,3,3,1,4,2,1],
    [1,6,2,7,1,0,5], [1,2,4,2,8,2,1]]), Vector([5,6,6,5]),
    x-> 3*x[4]  , Vector(7,-100), Vector(7,100)); "

                                  [  -18]
                                  [     ]
                                  [   50]
                                  [     ]
                                  [  -39]
                                  [     ]
                                  [-100], -300, 5.130
                                  [     ]
                                  [   47]
                                  [     ]
                                  [  -94]
                                  [     ]
                                  [   91]

"19.1,Graver(Matrix([[1,0,1,4,2,6,0],[2,4,7,1,2,4,1],[4,2,9,0,4,2,4],
    [5,1,2,3,2,0,2], [6,3,0,2,1,5,1]]), Vector([5,7,10,6,7]),
    x-> x[3]^2  , Vector(7,-100), Vector(7,100));"

                                    [-13]
                                    [    ]
                                    [ 10]
                                    [    ]
                                    [ -4]
                                    [    ]
                                    [ 11], 16, 0.160
                                    [    ]
                                    [-20]
                                    [    ]
                                    [  3]
                                    [    ]
                                    [ 38]

"19.2,Graver(Matrix([[1,0,1,4,2,6,0],[2,4,7,1,2,4,1],[4,2,9,0,4,2,4],
    [5,1,2,3,2,0,2], [6,3,0,2,1,5,1]]), Vector([5,7,10,6,7]),
    x-> 4*x[3]-x[6]  , Vector(7,-100), Vector(7,100));"

                                   [-49]
                                   [    ]
                                   [ 73]
                                   [    ]
                                   [-52]
                                   [    ]
                                   [  2], -196, 0.160
                                   [    ]
                                   [ 85]
                                   [    ]
                                   [-12]
```

```
                                        [     ]
                                        [  53]

"20.1,Graver(Matrix([[4,1,6,5,1,2,0,8,1],[4,4,1,2,1,0,5,7,2],[
    4,5,8,11,2,10,1,3,2], [4,2,6,0,5,7,3,4],
    [1,2,0,1,2,0,1,2]]), Vector([5,4,8,2,3]), x->
    x[5]^2+x[8]^2  , Vector(9,-100), Vector(9,100)); "

Error, (in Groebner:-F4:-GroebnerBasis) Maple was unable to
    allocate enough memory to complete this computation.
    Please see ?alloc

"20.2,Graver(Matrix([[4,1,6,5,1,2,0,8,1],[4,4,1,2,1,0,5,7,2],
    [4,5,8,11,2,10,1,3,2], [4,2,6,0,5,7,3,4],
    [1,2,0,1,2,0,1,2]]), Vector([5,4,8,2,3]), x-> x[5]  ,
    Vector(9,-100), Vector(9,100)); "

Error, (in Groebner:-F4:-GroebnerBasis) Maple was unable to
    allocate enough memory to complete this computation.
    Please see ?alloc

"21.1,
    Graver(Matrix([[1,4,5,2,1,2,0,8,1],[3,2,4,0,6,1,9,0,2],
    [4,0,3,1,2,5,6,4,1], [1,5,1,9,10,1,3,2],
    [1,5,7,1,8,0,6,2]]), Vector([5,9,14,2,3]), x-> x[5]^2  ,
    Vector(9,-100), Vector(9,100)); ");

Error, (in Groebner:-F4:-GroebnerBasis) Maple was unable to
    allocate enough memory to complete this computation.
    Please see ?alloc

"21.2,
    Graver(Matrix([[1,4,5,2,1,2,0,8,1],[3,2,4,0,6,1,9,0,2],
    [4,0,3,1,2,5,6,4,1], [1,5,1,9,10,1,3,2],
    [1,5,7,1,8,0,6,2]]), Vector([5,9,14,2,3]), x-> x[5]+x[7],
    Vector(9,-100), Vector(9,100)); ");

Error, (in Groebner:-F4:-GroebnerBasis) Maple was unable to
    allocate enough memory to complete this computation.
    Please see ?alloc


"24.1,Graver(Matrix([[5,2,1,3,0,3,2],[1,3,2,5,3,0,1],
    [1,1,2,3,1,0,4], [1,2,8,5,0,2,1]]), Vector([3,5,8,5]),
    x-> x[1]^2+x[5]^2, Vector(7,-100), Vector(7,100));"

                                         [   0]
                                         [    ]
                                         [-23]
                                         [    ]
                                         [  -5]
                                         [    ]
                                         [  18], 1, 2.219
                                         [    ]
```

```
                                   [ −1]
                                   [   ]
                                   [  2]
                                   [   ]
                                   [ −3]

"24.2,Graver(Matrix([[5,2,1,3,0,3,2],[1,3,2,5,3,0,1],
    [1,1,2,3,1,0,4], [1,2,8,5,0,2,1]]), Vector([3,5,8,5]),
    x-> x[1]+8*x[5]^2, Vector(7,−100), Vector(7,100));"

                                [−100]
                                [    ]
                                [  94]
                                [    ]
                                [ −56]
                                [    ]
                                [  42], −900, 2.380
                                [    ]
                                [−100]
                                [    ]
                                [  65]
                                [    ]
                                [  25]

"25.1,Graver(Matrix([[3,1,5,2,8,1,2],[6,2,1,3,6,1,2],
    [1,8,3,1,4,2,1], [1,4,2,7,1,0,5]]), Vector([4,5,6,6]),
    x-> x[3]^2, Vector(7,−100), Vector(7,100));, 1kwad"

Error, (in Groebner:−F4:−GroebnerBasis) Maple was unable to
    allocate enough memory to complete this computation.
    Please see ?alloc

"25.2,Graver(Matrix([[3,1,5,2,8,1,2],[6,2,1,3,6,1,2],
    [1,8,3,1,4,2,1], [1,4,2,7,1,0,5]]), Vector([4,5,6,6]),
    x-> x[3]+3*x[6], Vector(7,−100), Vector(7,100));"

Error, (in Groebner:−F4:−GroebnerBasis) Maple was unable to
    allocate enough memory to complete this computation.
    Please see ?alloc

"26.1,Graver(Matrix([[2,1,5,2,1,0,2],[1,2,1,1,6,1,2],
    [1,3,2,1,4,2,1], [4,1,1,0,1,2,2]]), Vector([2,1,3,4]),
    x-> x[1]^2, Vector(7,−100), Vector(7,100));, 1kwad"

                                   [ 0]
                                   [  ]
                                   [ 3]
                                   [  ]
                                   [ 0]
                                   [  ]
                                   [−2], 0, 1.530
                                   [  ]
                                   [−1]
                                   [  ]
```

```
                    [-1]
                    [  ]
                    [ 2]


"26.2,Graver(Matrix([[2,1,5,2,1,0,2],[1,2,1,1,6,1,2],
    [1,3,2,1,4,2,1], [4,1,1,0,1,2,2]]), Vector([2,1,3,4]),
    x-> x[1]+x[2]+x[3], Vector(7,-100), Vector(7,100));"

                    [-71]
                    [   ]
                    [-36]
                    [   ]
                    [-36]
                    [   ]
                    [100], -143, 2.510
                    [   ]
                    [-36]
                    [   ]
                    [100]
                    [   ]
                    [ 98]


"27.1,Graver(Matrix([[5,1,1,3,0,1,2],[1,3,2,5,4,0,1],
    [1,4,2,3,1,0,5], [1,2,3,5,0,2,1]]), Vector([3,5,3,6]),
    x-> x[5]^2, Vector(7,-100), Vector(7,100));"

                    [-31]
                    [   ]
                    [-68]
                    [   ]
                    [-67]
                    [   ]
                    [ 65], 0, 3.200
                    [   ]
                    [  0]
                    [   ]
                    [  0]
                    [   ]
                    [ 49]

"27.2,Graver(Matrix([[5,1,1,3,0,1,2],[1,3,2,5,4,0,1],
    [1,4,2,3,1,0,5], [1,2,3,5,0,2,1]]), Vector([3,5,3,6]),
    x-> 3*x[5]-x[6], Vector(7,-100), Vector(7,100));"

                    [    4]
                    [     ]
                    [  100]
                    [     ]
                    [  -91]
                    [     ]
                    [   68], -194, 8.250
                    [     ]
```

```
                              [  -98]
                              [     ]
                              [-100]
                              [     ]
                              [  -65]

"28.1,Graver(Matrix([[4,1,5,2,8,1,2],[7,2,0,3,6,1,2],[1,6,3,1,4,2,7],
    [0,3,1,2,7,1,9], [1,2,3,4,6,2,7]]), Vector([9,5,7,6,5]),
    x-> x[2]^2, Vector(7,-100), Vector(7,100));"

                                [-15]
                                [    ]
                                [  0]
                                [    ]
                                [-19]
                                [    ]
                                [-14],  0,  0.200
                                [    ]
                                [ 20]
                                [    ]
                                [ 66]
                                [    ]
                                [-17]

"28.2,Graver(Matrix([[4,1,5,2,8,1,2],[7,2,0,3,6,1,2],[1,6,3,1,4,2,7],
    [0,3,1,2,7,1,9], [1,2,3,4,6,2,7]]), Vector([9,5,7,6,5]),
    x-> x[2]+6*x[3], Vector(7,-100), Vector(7,100));"

                               [-44]
                               [    ]
                               [ 23]
                               [    ]
                               [-39]
                               [    ]
                               [  0],  -211,  0.180
                               [    ]
                               [ 45]
                               [    ]
                               [ 93]
                               [    ]
                               [-48]

"29.1,Graver(Matrix([[2,1,5,2,8,1,2],[6,2,1,3,6,1,2],
    [1,8,3,1,4,2,1], [1,5,2,7,1,0,5]]), Vector([4,5,6,6]),
    x-> x[3]^2+x[6]^2, Vector(7,-100), Vector(7,100));"

                                [ 2]
                                [   ]
                                [ 1]
                                [   ]
                                [ 2]
                                [   ]
                                [-2],  13,  2.460
                                [   ]
```

```
                        [−1]
                        [  ]
                        [−3]
                        [  ]
                        [ 2]

"29.2,Graver(Matrix([[2,1,5,2,8,1,2],[6,2,1,3,6,1,2],[1,8,3,1,4,2,1],
    [1,5,2,7,1,0,5]]), Vector([4,5,6,6]), x−> x[3]+x[6],
    Vector(7,−100), Vector(7,100));"

                      [  −63]
                      [     ]
                      [   26]
                      [     ]
                      [−100]
                      [     ]
                      [   −5], −194, 2.400
                      [     ]
                      [   84]
                      [     ]
                      [  −94]
                      [     ]
                      [   18]

"30.1,Graver(Matrix([[2,1,5,2,1,0,2],[1,2,0,1,6,1,2],[1,3,2,1,4,2,1],
    [4,3,1,0,1,2,2]]), Vector([2,1,3,4]), x−> x[1]^2+x[3]^2,
    Vector(7,−100), Vector(7,100));"

                         [ 1]
                         [  ]
                         [ 4]
                         [  ]
                         [ 0]
                         [  ]
                         [ 0], 1, 1.650
                         [  ]
                         [ 0]
                         [  ]
                         [−4]
                         [  ]
                         [−2]

"30.2,Graver(Matrix([[2,1,5,2,1,0,2],[1,2,0,1,6,1,2],[1,3,2,1,4,2,1],
    [4,3,1,0,1,2,2]]), Vector([2,1,3,4]), x−> x[7]−x[6],
    Vector(7,−100), Vector(7,100));"

                      [   83]
                      [     ]
                      [  −96]
                      [     ]
                      [  −18]
                      [     ]
                      [  100], −178, 1.710
                      [     ]
```

```
                                        [   22]
                                        [     ]
                                        [   78]
                                        [     ]
                                        [-100]

"31.1,Graver(Matrix([[4,1,5,2,8,1,9],[7,2,0,3,6,1,8],[5,6,3,1,4,2,7],
   [3,11,2,7,1,9]]), Vector([9,5,7,6]), x-> x[2]^2+x[4]^2,
   Vector(7,-100), Vector(7,100));"

                                        [ 8]
                                        [  ]
                                        [ 1]
                                        [  ]
                                        [ 6]
                                        [  ]
                                        [ 0], 1, 13.360
                                        [  ]
                                        [ 4]
                                        [  ]
                                        [-5]
                                        [  ]
                                        [-9]

"31.2,Graver(Matrix([[4,1,5,2,8,1,9],[7,2,0,3,6,1,8],
   [5,6,3,1,4,2,7], [3,11,2,7,1,9]]), Vector([9,5,7,6]), x->
   2*x[2]-3*x[4], Vector(7,-100), Vector(7,100));"

                                        [ 49]
                                        [   ]
                                        [-13]
                                        [   ]
                                        [ 84]
                                        [   ]
                                        [ 99], -323, 6.170
                                        [   ]
                                        [-94]
                                        [   ]
                                        [-85]
                                        [   ]
                                        [  5]
```