SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

The Buchberger-Möller Algorithm

av

Jonas Klang

2012 - No 24

The Buchberger-Möller Algorithm

Jonas Klang

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Samuel Lundqvist

2012

Abstract

In this thesis we will describe the Buchberger-Möller-algorithm and also implement it using the computer programming language C++. The algorithm is used to calculate a vector space basis for a certain type of quotient ring. We will also go through all the mathematical theory behind the implementation.

Acknowledgements

I would like to thank my supervisor Samuel Lundqvist who's given me excellent help and support through all the stages of this thesis.

1 Introduction

This thesis is about the the Buchberger-Möller-algorithm and also about implementing it with support for ordering described by matrices, using the programming language C++. An implementation of the algorithm in C++ has not been done before. The Buchberger-Möller-algorithm is of interest in many science areas such as coding theory, interpolation problems, statistics and molecular biology [5]. In order to understand how the algorithm works we need a foundation of theory which is what we start this paper describing.

We go through definitions of fields, rings, monomials, ideals, linear independence, vector space basis, order ideal of monomials and we specify three types of sorting orders for monomials that we'll be using when implementing the Buchberger-Möller-algorithm. We also list three matrices with which we can always order monomials by multiplying these matrices with the monomials exponent vectors. We also prove that these matrices always work for this purpose. We also include a few proofs and examples of the different theory pieces we are describing.

In Section 2 we describe the actual Buchberger-Möller-algorithm and show two examples of it being used. We finish the section with some more theory and describe how the algorithm can be used for finding a Gröbner basis.

The final Section talks about the actual program written to implement the Buchberger-Möller-algorithm. We go through a few of the functions that make up the program and we also run the program using different data and list the results in a table showing the efficiency of the program.

The paper ends with a bibliography of the literature being used as inspiration for this paper.

2 Theory

This section will serve as an introduction to the theory needed to understand the contents of this paper. Here we will go through mathematical terms such as monomials, polynomials, fields, ideals, vanishing ideals, vector basis and matrix rank.

We begin with defining a field.

Definition 1. *Let F be a set on which the two binary operations; addition and multiplication, are defined and denoted by $+$ and $\cdot$ respectively. Then F is called a field if for all elements in F the following properties hold:*

(i) Closure of F under addition and multiplication: For all elements $a, b \in F$, the sum $a + b$ and the product $a \cdot b$ are both well-defined elements of F.

(ii) Associativity: For all a, b and $c \in F$,

$$a + (b + c) = (a + b) + c \text{ and } a \cdot (b \cdot c) = (a \cdot b) \cdot c. \tag{1}$$

(iii) Commutativity: For all $a, b \in F$,

$$a + b = b + a \text{ and } a \cdot b = b \cdot a. \tag{2}$$

(iv)Distributivity: For all a, b and $c \in F$,

$$a \cdot (b + c) = a \cdot b + a \cdot c. \tag{3}$$

(v) Additive and multiplicative identity elements: There exists an additive identity element called 0, such that for all $a \in F$,

$$a + 0 = a. \tag{4}$$

And there exists a multiplicative identity element called 1, such that for all $a \in F$,

$$a \cdot 1 = a. \tag{5}$$

(vi)Additive and multiplicative inverses: There exists an additive inverse element called -a, such that for all $a \in F$,

$$a + (-a) = 0. \tag{6}$$

And for $a \neq 0$ there exists a multiplicative inverse element called a^{-1}, such that for all $a \in F$,

$$a \cdot a^{-1} = 1. \tag{7}$$

Definition 2. *A commutative ring is defined the same way as a field but it lacks the multiplicative inverse.*

Example 1. *Z_p, where p is a prime number, is a field. See [1] for proof.*

Let $\Bbbk$ be a field. The polynomial ring $\Bbbk[x_1, ..., x_n]$ consists of all polynomials with coefficients in $\Bbbk$. A monomial is defined as an element $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ and a polynomial can then be regarded as a finite linear combination of monomials.

Example 2. *The polynomial* $3x_1 + 2x_1x_2 + 2x_2 \in Q[x_1, x_2]$ *is a linear combination of the monomials* x_1, x_1x_2 *and* x_2.

It should also be noted that 1 is a monomial of the form $x_1^0 \cdots x_n^0$.

Theorem 1. *Lek* $\Bbbk$ *be a field. Then* $\Bbbk[x_1, ..., x_n]$ *is a ring.*

Proof. *An easy check of all the ring axioms shows that they are satisfied.*

Definition 3. *Let* R *be commutative ring. A nonempty subset* I *of* R *is called an ideal of* R *if the following holds:*
(i) $a \pm b \in I$ *for all* a,$b \in I$.
(ii) $ra \in I$ *for all* $a \in I$ *and* $r \in R$.

An ideal in $\Bbbk[x_1, ..., x_n]$ is finitely generated, which means that there exists a finite numbers of polynomials $(f_1, ..., f_r)$, such that for any $a \in I \subseteq \Bbbk[x_1, ..., x_n]$, $a = h_1f_1 + ... + h_rf_r$ for $h_i \in \Bbbk[x_1, ..., x_n]$ and $f_i \in I$.

A point in $\Bbbk^n$ in an n-tuple of n elements with coefficients in $\Bbbk$. If we have $n = 3$ and $\Bbbk = Z_2$, then $p_1 = (1, 1, 0)$ and $p_2 = (1, 0, 1)$ are examples of points.

Theorem 2. *Let* I *be an ideal in the commutative ring* R. *Then* R/I *is also a commutative ring, called a quotient ring, such that for all* $a, b \in R$

$$(a + I) + (b + I) = (a + b) + I$$

and

$$(a + I) \cdot (b + I) = ab + I$$

Proof. *The proof can be found in [1]*

Definition 4. *Let* $P = \{p_1, ..., p_m\}$ *where* $p_i \in \Bbbk^n$. *The set of elements* $\{f\}$ *where* $f \in \Bbbk[x_1, ..., x_n]$ *such that* $f(p_1) = ... = f(p_m) = 0$ *is called the vanishing ideal of* P *and is denoted by* $I(P)$.

Theorem 3. *Let* $P = \{p_1, ..., p_m\}$ *where* $p_i \in \Bbbk^n$. *Then* $I(P)$ *is an ideal in* $\Bbbk[x_1, ..., x_n]$.

Proof. *We start by proving that criterion (i) holds. For any elements* $f, g \in I(P)$ *we get the following:*

$$(f + g)(p_i) = f(p_i) + g(p_i) = 0 + 0 = 0, \tag{8}$$

hence $f + g \in I(P)$. *In the same way we see that criterion (ii) also holds for any element* $h \in \Bbbk[x_1, ..., x_n]$*:*

$$h \cdot f(p_i) = h(p_i) \cdot f(p_i) = h(p_i) \cdot 0 = 0, \tag{9}$$

hence $h \cdot f \in I(P)$. And since both criterions hold we have proven that I(P) is an ideal.

One application of the Buchberger-Möller-algorithm is that it can be used to compute a set of generators for I(P).

Monomial ordering

Being able to put a list of monomials in order is very useful, and often essential, when handling monomials in computer algebra. A monomial order respects multiplication ($m_1 > m_2 \Longrightarrow xm_1 > xm_2$) and 1 is the smallest monomial. We will use three ways to order monomials that we will now define.

Definition 5. *(Lexicographical order)*
Let $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$ and $\beta = (\beta_1, \beta_2, ...\beta_n) \in \Bbbk[x_1, ..., x_n]$. Then we say $\alpha > \beta$ lexicographically if, in the vector difference $\alpha - \beta \in \Bbbk$, the left-most nonzero entry is positive. Meaning $x^\alpha > x^\beta$ lexicographically if $\alpha > \beta$ lexicographically. We will sometimes refer to Lexicographical order as Lex.

Definition 6. *(Degree Lexicographical Order)*
Let $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$ and $\beta = (\beta_1, \beta_2, ...\beta_n) \in \Bbbk[x_1, ..., x_n]$. Then we say $\alpha > \beta$ in the Degree Lexicographical Order if
$|\alpha| = \sum_{i=1}^{n} \alpha_i > \sum_{i=1}^{n} \beta_i = |\beta|$ or, $|\alpha| = |\beta|$ and $\alpha > \beta$ lexicographically. We will sometimes refer to Degree Lexicographical order as DegLex.

Definition 7. *(Degree Reverse Lexicographical Order)*
Let $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$ and $\beta = (\beta_1, \beta_2, ...\beta_n) \in \Bbbk[x_1, ..., x_n]$. Then we say $\alpha > \beta$ in the Degree Reverse Lexicographical Order if
$|\alpha| = \sum_{i=1}^{n} \alpha_i > \sum_{i=1}^{n} \beta_i = |\beta|$ or, $|\alpha| = |\beta|$ and in $\alpha - \beta$, the rightmost nonzero entry is negative. We will sometimes refer to Degree Reverse Lexicographical order as DegRevLex.

Example 3. *Consider the following monomials:*

$$x_1^2, x_2^2, x_1x_2x_3, x_1^2x_2, x_3, x_1x_3^2, x_1x_3.$$

The exponent vectors for these monomials are:

$$(2,0,0), (0,2,0), (1,1,1), (2,1,0), (0,0,1), (1,0,2), (1,0,1).$$

If we order the monomials lexicographically we get:

$$(2,1,0) > (2,0,0) > (1,1,1) > (1,0,2) > (1,0,1) > (0,2,0) > (0,0,1),$$

or

$$x_1^2x_2 > x_1^2 > x_1x_2x_3 > x_1x_3^2 > x_1x_3 > x_2^2 > x_3.$$

If we order the monomials using DegLex we get:

$$(2,1,0) > (1,1,1) > (1,0,2) > (2,0,0) > (1,0,1) > (0,2,0) > (0,0,1),$$

or

$$x_1^2x_2 > x_1x_2x_3 > x_1x_3^2 > x_1^2 > x_1x_3 > x_2^2 > x_3.$$

And if we order the monomials using DegRevLex we get:

$$(2,1,0) > (1,1,1) > (1,0,2) > (2,0,0) > (0,2,0) > (1,0,1) > (0,0,1),$$

or

$$x_1^2x_2 > x_1x_2x_3 > x_1x_3^2 > x_1^2 > x_2^2 > x_1x_3 > x_3.$$

The following lemma was proved by Robbiano. [7]

Lemma 1. *For a fixed degree of the biggest included monomial in a finite set every monomial order $\prec$ can be described using an integer $n \times n$ - matrix M. The order between two monomials x^α and x^β can be decided by comparing the two vectors $M\alpha$ and $M\beta$ lexicographically.*

Example 4. *Let us use the same monomials as in the previous example. We will use the following matrices to multiply with:*

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; J = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \text{ and } K = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

We multiply the monomials with the identity matrix I and we get:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} > \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} >$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} > \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} >$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} > \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} >$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

And we see that multiplying with I in this example is equivalent to comparing the monomials lexicographically.
When we multiply the monomials with the J matrix we get:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} > \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} >$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix} > \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} >$$
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} > \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix} >$$
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

And we see that multiplying with J in this example is equivalent to comparing the monomials using DegLex.
And if we instead multiply the monomials with the K matrix we get:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ -1 \end{bmatrix} > \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \\ -1 \end{bmatrix} >$$
$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \\ 0 \end{bmatrix} > \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} >$$
$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ -2 \end{bmatrix} > \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix} >$$
$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$$

And we see that multiplying with K in this example is equivalent to comparing the monomials using DegRevLex

We will now prove that multiplying with these matrices does not just work with these examples but they work with all monomials.

Theorem 4. *The lexicographical order can be described using the identity matrix.*

Proof. *Comparing α and β lexicographically is equivalent to comparing $I\alpha$ and $I\beta$, where I is the identity matrix, lexicographically.*

Theorem 5. *$x^\alpha >_{degLex} x^\beta$ if and only if $M\alpha >_{lex} M\beta$ where*

$$M = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}$$

Proof. *We must show that* $x^\alpha >_{degLex} x^\beta$ *is equivalent to* $M\alpha >_{lex} M\beta$. *Suppose that* $x^\alpha >_{degLex} x^\beta$. *Then if* $\sum_{i=1}^n \alpha_i > \sum_{i=1}^n \beta_i$, *then clearly* $M\alpha >_{lex} M\beta$. *If* $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$, *let* j *be the least index such that* $\alpha_j > \beta_j$ *(Clearly such an index exists since we assume that* $x^\alpha >_{degLex} x^\beta$*). Then* $\alpha_1 = \beta_1, \alpha_2 = \beta_2, ..., \alpha_{j-1} = \beta_{j-1}, \alpha_j > \beta_j$. *Notice that* $j < n$ *(For if* $j = n$, *then,* $\alpha_1 = \beta_1, \alpha_2 = \beta_2, ..., \alpha_{n-1} = \beta_{n-1}, \alpha_n > \beta_n$, *which contradicts the assumption* $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$*). We set*

$$M\alpha - M\beta = \begin{bmatrix} \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \beta_i \\ \alpha_1 - \beta_1 \\ \vdots \\ \alpha_j - \beta_j \\ \vdots \\ \alpha_{n-1} - \beta_{n-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \alpha_j - \beta_j \\ \vdots \\ \alpha_{n-1} - \beta_{n-1} \end{bmatrix}$$

and since we assume that $\alpha_j > \beta_j$, *it follows that* $M\alpha >_{lex} M\beta$.

Suppose that $M\alpha >_{lex} M\beta$. *Let* j *be the first index where* $(M\alpha)_j > (M\beta)_j$. *Hence* $(M\alpha)_1 = (M\beta)_1, (M\alpha)_2 = (M\beta)_2, ..., (M\alpha)_{j-1} = (M\beta)_{j-1}, (M\alpha)_j > (M\beta)_j$. *If* $j = 1$, *then* $(M\alpha)_1 > (M\beta)_1$ *which is equivalent to* $\sum_{i=1}^n \alpha_i > \sum_{i=1}^n \beta_i$, *thus* $x^\alpha >_{degLex} x^\beta$. *If* $j > 1$ *then* $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i, \alpha_1 = \beta_1, \alpha_2 = \beta_2, ..., \alpha_{j-1} = \beta_{j-1}, \alpha_j > \beta_j$. *Hence* $x^\alpha >_{degLex} x^\beta$. *The proof is complete.*

Theorem 6. $x^\alpha >_{degRevLex} x^\beta$ *if and only if* $M\alpha >_{lex} M\beta$ *where*

$$M = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & 0 & 0 & \dots & -1 \\ 0 & 0 & \dots & -1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & -1 & \dots & 0 & 0 \end{bmatrix}$$

Proof. *We must show that* $x^\alpha >_{degRevLex} x^\beta$ *is equivalent to* $M\alpha >_{lex} M\beta$. *Suppose that* $x^\alpha >_{degRevLex} x^\beta$. *Then if* $\sum_{i=1}^n \alpha_i > \sum_{i=1}^n \beta_i$, *then clearly* $M\alpha >_{lex} M\beta$. *If* $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$, *let* j *be the biggest index such that* $\alpha_j < \beta_j$ *(Clearly such an index exists since we assume that* $x^\alpha >_{degRevLex} x^\beta$*). Then* $\alpha_n = \beta_n, \alpha_{n-1} = \beta_{n-1}, ..., \alpha_{j+1} = \beta_{j+1}, \alpha_j < \beta_j$. *Notice that* $j \geq 2$ *(For if* $j = 1$, *then,* $\alpha_n = \beta_n, \alpha_{n-1} = \beta_{n-1}, ..., \alpha_2 = \beta_2, \alpha_1 < \beta_1$, *which contradicts the assumption* $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$*). We set*

$$M\alpha - M\beta = \begin{bmatrix} \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \beta_i \\ -\alpha_n - (-\beta_n) \\ -\alpha_{n-1} - (-\beta_{n-1}) \\ \vdots \\ -\alpha_j - (-\beta_j) \\ \vdots \\ -\alpha_2 - (-\beta_2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -\alpha_j + \beta_j \\ \vdots \\ -\alpha_2 + \beta_2 \end{bmatrix}$$

and since we assume that $\alpha_j < \beta_j$, it follows that $M\alpha >_{lex} M\beta$.

Suppose that $M\alpha >_{lex} M\beta$. Let j be the least index where $(M\alpha)_j > (M\beta)_j$. Hence $(M\alpha)_1 = (M\beta)_1, (M\alpha)_2 = (M\beta)_2, ..., (M\alpha)_{j-1} = (M\beta)_{j-1}, (M\alpha)_j > (M\beta)_j$. If $j = 1$, then $(M\alpha)_1 > (M\beta)_1$ which is equivalent to $\sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \beta_i$, thus $x^\alpha >_{degRevLex} x^\beta$. If $j > 1$ then $\sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \beta_i, \alpha_n = \beta_n, \alpha_{n-1} = \beta_{n-1}, ..., \alpha_{j+1} = \beta_{j+1}, \alpha_j < \beta_j$. Hence $x^\alpha >_{degRevLex} x^\beta$. The proof is complete.

Linear dependency

A vector is linearly independent from other vectors if it cant be written as a linear combination of these. In the same way a vector is linearly dependent of a set of vectors if it can be written as a linear combination of the set of vectors.

A vector basis over a field $\Bbbk$ is a set of linearly independent vectors that together can be used in a linear combination to create any other vector in a given vector space. The produced vector has a linear combination of vectors that is unique. The number of vectors in a vector basis is called the dimension of the vector space.

The following theorem is fundamental:

Theorem 7. *Let $P = \{p_1, p_2, \cdots, p_m\}$ be a set of m points. Then*

$$\Bbbk[x_1, ..., x_n]/I(P), \tag{10}$$

is of finite dimension as a vector space over $\Bbbk$ and the dimension is equal to the number of points m.

Proof. *The proof is beyond the scope of this thesis. But it can be found in [4]*

We will now examine the consequences of the theorem:

Let $\{e_1, ..., e_m\}$ be polynomials in $\Bbbk[x_1, ..., x_n]$ and suppose that $[e_1], ..., [e_n]$ is a vector space basis for $\Bbbk[x_1, ..., x_n]/I(P)$, where we by $[e_i]$ mean the residue class of $I(P)$ containing e_i. This means that if we fix the e_is then any polynomial $f \in \Bbbk[x_1, ..., x_n]$ can be written uniquely as

$$[f] = c_1[e_1] + \cdots + c_m[e_m], \tag{11}$$

or

$$[f] - c_1[e_1] - \cdots - c_m[e_m] = 0, \tag{12}$$

which is the same as

$$f - c_1e_1 - \cdots - c_me_m \in I(P), \tag{13}$$

this in turn means that $f - c_1e_1 - \cdots - c_me_m$ is zero at every point, that is

$$(f - c_1e_1 - \cdots - c_me_m)(p_1) = \cdots = (f - c_1e_1 - \cdots - c_me_m)(p_m) = 0, \tag{14}$$

An Order ideal of monomials, which we call an OIM, is a set of monomials that is closed under taking submonomials. By a submonomial we mean that $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ is a submonomial to $x_1^{\beta_1} \cdots x_n^{\beta_n}$ iff $\alpha_i \leq \beta_i \ \forall i$

Example 5. *$\{1, x_1, x_2, x_1x_2\}$ is an OIM but $\{1, x_1x_2, x_2\}$ is not since x_1 is a sub monomial of x_1x_2 which is not part of the set.*

Theorem 8. *If P is a set of points, then there is always an $OIM = e_1, e_2, ..., e_m$ such that $[e_1], [e_2], ..., [e_m]$ forms a vector space basis for $\Bbbk[x_1, ..., x_m]/I(P)$.*

Proof. *This follows from the Buchberger-Möller algorithm in the next chapter.*

3 The Buchberger-Möller algorithm

The Buchberger-Möller algorithm, originally described in [2], is designed to, given points P = $\{p_1, ..., p_m\}$ in a field $\Bbbk$, calculate a Gröbner basis of I(P) and a vector space basis for $\Bbbk[x_1, ..., x_n]/I(P)$, where I(P) is the vanishing ideal for the given points. We will focus on the vector space basis. The monomials making up the vector space basis is an OIM. The key of the Buchberger-Möller algorithm, is the fact that $\{[e_1], \ldots, [e_m]\}$ is a vector space basis for $\Bbbk[x_1, \ldots, x_n]/I(P)$ if and only if the vectors $e_1(P), \ldots, e_m(P)$ are linearly independent.

We will use the following notations to present the Buchberger-Möller-algorithm: P = $\{p_1, ..., p_m\}$ are the points we will use. L is a list of monomials, sorted in an increasing fixed monomial order which we denote as $\prec$. B = $e_1, e_2, ...$ is a list of sorted monomials such that $[e_1], [e_2], ...$ is linearly independent in $\Bbbk[x_1, ..., x_n]/I(P)$. This is at the end of the algorithm going to constitute our vector space basis. We have one more list of monomials; G, which is also sorted in an increasing order. G is a subset to the border, see Definition 8.

When we write e(P) we mean the vector $(e(p_1), ..., e(p_m))$. When we write B(P) we mean the matrix $(e_i(p_j))_{ij}$.

1. L = {1}, B = (), G = ()

2. If $|B|$ = m, return B. Else, clear the list L from multiples of elements in G, meaning elements in L that are divisible by G, and let e = First[L], L = Rest[L].

3. If e(P) is linearly dependent with respect to the rows in B(P), set G = G $\cup$ e and go back to step 2.

4. If e(P) is linearly independent with respect to the rows in B(P), set B = B $\cup$ e and L = merge(L,$(x_n e, ..., x_1 e)$) and go back to step 2.

Remark: By merge we mean that we take two sorted lists and we merge them together into one sorted list. And since we in our order assume that $x_n < x_{n-1} < ... < x_1$ then we get that the lists to be merged have that order and hence the list L is merged with the list $x_n e < x_{n-1} e < ... < x_1 e$.

Example 6. *Let* $P = ((1,1,1),(0,0,1),(0,1,1))$ *be our points in* Z_2^3 *and let* $\prec$ *be the lexicographical order with* $x_1 \succ x_2 \succ x_3$*. In step one* $L = (1)$ *and in step two G is empty so we get* $e = 1$ *and* $L = ()$*. In step three we get* $e(P) = (1,1,1)$ *and since B is empty e(P) is linearly independent with respect to the rows of B(P) and we get* $B = (1)$ *and* $L = (x_3, x_2, x_1)$ *in step four. Back in step*

two we G is empty so we get $e = x_3$ *and* $L = (x_2, x_1)$*. In step three we get* $e(P) = x_3(P) = (1, 1, 1)$ *and since clearly this is linearly dependent with respect to B(P) since they are the same and we get* $G = (x_3)$*. Back in step two non of the elements in L is a multiple of* x_3 *so* $e = x_2$ *and* $L = (x_1)$*. In step three we get* $e(P) = x_2(P) = (1, 0, 1)$ *which is linearly independent with respect to the rows of B(P) so in step four we get* $B = (1, x_2)$ *and* $L = (x_3x_2, x_2^2, x_1, x_2x_1)$*. Back to step two we see that* x_3x_2 *is a multiple of G so we remove that element and we get get* $e = x_2^2$ *and* $L = (x_1, x_2x_1)$*. In step three we get* $x_2^2(P) = (1, 0, 1) = x_2(P)$*, so we get* $G = (x_3, x_2^2)$*. Back in step two there are no multiples of G in L so we get* $e = x_1$ *and* $L = (x_2x_1)$*. In step three we get* $x_1(P) = (1, 0, 0)$ *which is linearly independent with respect to the rows in B(P) and in step 4 we get* $B = (1, x_2, x_1)$ *and* $L = (x_3x_1, x_2x_1, x_1^2)$*. Back in step 2 we see that* $|B| = 3$ *and the algorithm terminates. This means that* $([1], [x_2], [x_1])$ *is a vector space basis for* $Z_2[x_1, x_2, x_3]/I(P)$*.*

Example 7. *Let*

$P = ((0, 0, 0, 0, 0), (1, 0, 0, 0, 0), (0, 1, 0, 0, 0), (1, 1, 0, 0, 0), (2, 1, 0, 0, 0))$ *be our points in* Z_3^5 *and let* $\prec$ *be the lexicographical order with* $x_1 \succ x_2 \succ x_3 \succ x_4 \succ x_5$*. In step one* $L = (1)$ *and in step two G is empty so we get* $e = 1$ *and* $L = ()$*. In step three we get* $e(P) = (1, 1, 1, 1, 1)$ *and since B is empty e(P) is linearly independent with respect to the rows of B(P) and we get* $B = (1)$ *and* $L = (x_5, x_4, x_3, x_2, x_1)$ *in step four. Back in step two there are no multiples of G so we get* $e = x_5$ *and* $L = (x_4, x_3, x_2, x_1)$*. In step three* $e(P) = x_5(P) = (0, 0, 0, 0, 0)$ *and this is linearly dependent with respect to the rows in B(P) since* $(1, 1, 1, 1, 1)$ *is in B(P) and* $0 \cdot (1, 1, 1, 1, 1) = (0, 0, 0, 0, 0)$*, therefore* $G = (x_5)$*. Back in step two there are no multiples of G so* $e = x_4$ *and* $L = (x_3, x_2, x_1)$*. In step three we then get* $e(P) = x_4(P) = (0, 0, 0, 0, 0)$ *and is therefore linearly dependent with respect to the rows in B(P) and we get* $G = (x_5, x_4)$*. Since again in step two there are no multiples of G in L we get* $e = x_3$ *and* $L = (x_2, x_1)$*. and once again in step three we get* $e(P) = x_3(P) = (0, 0, 0, 0, 0)$ *which is linearly dependent with respect to the rows in B(P) and we get* $G = (x_5, x_4, x_3)$*. In step two there are no multiples of G and we get* $e = x_2$ *and* $L = (x_1)$*. In step three we get* $e(P) = x_2(P) = (0, 0, 1, 1, 1)$ *which is linearly independent with respect to the rows in B(P) and in step four we get* $B = (1, x_2)$ *and* $L = (x_5x_2, x_4x_2, x_3x_2, x_2^2, x_1, x_2x_1)$*. In step two we remove the multiples of G from L and we get* $e = x_2^2$ *and* $L = (x_1, x_2x_1)$*. In step three we then get* $e(P) = x_2^2(P) = (0, 0, 1, 1, 1) = x_2(P)$ *and is therefore linearly dependent with respect to the rows in B(P) and we get* $G = (x_5, x_4, x_3, x_2^2)$*. In step two we remove* x_2^2 *from L and we get* $e = x_1$ *and* $L = (x_1x_2)$*. So in step three we get* $e(P) = x_1(P) = (0, 1, 0, 1, 2)$ *which is is linearly independent with respect to the rows of B(P) and we get* $B = (1, x_2, x_1)$ *and* $L = (x_5x_1, x_4x_1, x_3x_1, x_2x_1, x_1^2)$*. We go back to step two and remove the multiples of G from L and we get* $e = x_1x_2$ *and* $L = (x_1^2)$*. In step 3 we get* $x_1x_2(P) = (0, 0, 0, 1, 2)$ *which is is linearly independent with respect to the rows of B(P) and in step four we get* $B = (1, x_2, x_1, x_2x_1)$ *and* $L = (x_5x_2x_1, x_4x_2x_1, x_3x_2x_1, x_2^2x_1, x_1^2, x_2x_1^2)$*. Once again we go back to step two and remove the multiples of G from L and we get*

$e = x_1^2$ *and* $L = (x_1^2 x_2)$*. In step 3 we get* $x_1^2(P) = (0, 1, 0, 1, 1)$ *which is is linearly independent with respect to the rows of B(P) and we get* $B = (1, x_2, x_1, x_2x_1, x_1^2)$ *and* $L = (x_5x_1^2, x_4x_1^2, x_3x_1^2, x_2x_1^2, x_1^3)$*. Finally back in step 2 we see that* $|B| = 5$ *and the algorithm terminates. This means that* $([1], [x_2], [x_1], [x_2x_1], [x_1^2])$ *is a vector space basis for* $Z_3[x_1, x_2, x_3, x_4, x_5]/I(P)$*.*

It is possible to implement the Buchberger-Möller-algorithm such that the number of arithmetic operations is $O(min(m, n) \cdot m^3)$. [6]

Definition 8. *(Border)*
Given an OIM, a border monomial is a monomial e_i *such that all of its sub-monomials belongs to the OIM, but* $e_i \notin OIM$*.*

Example 8. *In Example 5 where the OIM is:* $1, x_2, x_1$*, the border monomials are:* x_1x_2, x_2^2, x_1^2*.*

Example 9. *In Example 6 where the OIM is:* $1, x_2, x_1, x_2x_1, x_1^2$*, the border monomials are:* $x_1^3, x_1^2x_2, x_2^2$*.*

Theorem 9. *If we chose a basis with the Buchberger-Möller algorithm and if* f_i *are border monomials, then* $(f_1 - \sum_j c_{ji}e_j, ..., f_k - \sum_j c_{ji}e_j)$ *is a Gröbner basis for I(P).*

Proof. *The proof is beyond the scope of this thesis, and since we will not define a Gröbner basis this theorem is an anecdote for people familiar with the theory.*

A Gröbner basis for an ideal is always a generator set for I, thus $I(P) = (f_1 - \sum_j c_{ji}e_j, ..., f_k - \sum_j c_{ji}e_j)$

Example 10. *In the previous example we concluded that the border monomials for Example 6 are* $x_1^3, x_1^2x_2, x_2^2$*. We can use these monomials and Theorem 8 to calculate a Gröbner basis for I(P).*

We start with the element x_1^3 *and start out to find* c_1, c_2, c_3, c_4 *and* c_5 *such that* $x_1^3 = c_1[1] + c_2[x_2] + c_3[x_1] + c_4[x_2x_1] + c_5[x_1^2] \in I(P)$*, or* $(x_1^3 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(P) = 0$*, and since* $P = (p_1, p_2, p_3, p_4, p_5) = ((0, 0, 0, 0, 0), (1, 0, 0, 0, 0), (0, 1, 0, 0, 0), (1, 1, 0, 0, 0), (2, 1, 0, 0, 0))$ *we have to solve the following equation systems:*

$$(x_1^3 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_1) = 0 \Rightarrow 0 - c_1 \cdot 1 - c_2 \cdot 0 - c_3 \cdot 0 - c_4 \cdot 0 - c_5 \cdot 0 = 0$$

$$(x_1^3 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_2) = 0 \Rightarrow 1 - c_1 \cdot 1 - c_2 \cdot 0 - c_3 \cdot 1 - c_4 \cdot 0 - c_5 \cdot 1 = 0$$

$$(x_1^3 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_3) = 0 \Rightarrow 0 - c_1 \cdot 1 - c_2 \cdot 1 - c_3 \cdot 0 - c_4 \cdot 0 - c_5 \cdot 0 = 0$$

$$(x_1^3 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_4) = 0 \Rightarrow 1 - c_1 \cdot 1 - c_2 \cdot 1 - c_3 \cdot 1 - c_4 \cdot 1 - c_5 \cdot 1 = 0$$

$$(x_1^3 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_5) = 0 \Rightarrow 2 - c_1 \cdot 1 - c_2 \cdot 1 - c_3 \cdot 2 - c_4 \cdot 2 - c_5 \cdot 1 = 0$$

When solving this equations system we get $c_1 = 0, c_2 = 0, c_3 = 1, c_4 = 0, c_5 = 0$*. Hence we can express* x_1^3 *as* $x_1^3 = x_1 \pmod{I(P)}$*.*

Now we find the c_1, c_2, c_3, c_4 and c_5 for $x_1^2x_2$ and we have to solve the following equation systems:

$$(x_1^2x_2 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_1) = 0 \Rightarrow 0 - c_1 \cdot 1 - c_2 \cdot 0 - c_3 \cdot 0 - c_4 \cdot 0 - c_5 \cdot 0 = 0$$

$$(x_1^2x_2 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_2) = 0 \Rightarrow 0 - c_1 \cdot 1 - c_2 \cdot 0 - c_3 \cdot 1 - c_4 \cdot 0 - c_5 \cdot 1 = 0$$

$$(x_1^2x_2 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_3) = 0 \Rightarrow 0 - c_1 \cdot 1 - c_2 \cdot 1 - c_3 \cdot 0 - c_4 \cdot 0 - c_5 \cdot 0 = 0$$

$$(x_1^2x_2 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_4) = 0 \Rightarrow 1 - c_1 \cdot 1 - c_2 \cdot 1 - c_3 \cdot 1 - c_4 \cdot 1 - c_5 \cdot 1 = 0$$

$$(x_1^2x_2 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_5) = 0 \Rightarrow 1 - c_1 \cdot 1 - c_2 \cdot 1 - c_3 \cdot 2 - c_4 \cdot 2 - c_5 \cdot 1 = 0$$

We get $c_1 = 0, c_2 = 0, c_3 = -1, c_4 = 1, c_5 = 1$. Hence we can express $x_1^2x_2$ as $x_1^2x_2 = -x_1 + x_2x_1 + x_1^2 \pmod{I(P)}$.

Finally we find the c_1, c_2, c_3, c_4 and c_5 for x_2^2 and we have to solve the following equation systems:

$$(x_2^2 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_1) = 0 \Rightarrow 0 - c_1 \cdot 1 - c_2 \cdot 0 - c_3 \cdot 0 - c_4 \cdot 0 - c_5 \cdot 0 = 0$$

$$(x_2^2 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_2) = 0 \Rightarrow 0 - c_1 \cdot 1 - c_2 \cdot 0 - c_3 \cdot 1 - c_4 \cdot 0 - c_5 \cdot 1 = 0$$

$$(x_2^2 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_3) = 0 \Rightarrow 1 - c_1 \cdot 1 - c_2 \cdot 1 - c_3 \cdot 0 - c_4 \cdot 0 - c_5 \cdot 0 = 0$$

$$(x_2^2 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_4) = 0 \Rightarrow 1 - c_1 \cdot 1 - c_2 \cdot 1 - c_3 \cdot 1 - c_4 \cdot 1 - c_5 \cdot 1 = 0$$

$$(x_2^2 - c_1 1 - c_2x_2 - c_3x_1 - c_4x_2x_1 - c_5x_1^2)(p_5) = 0 \Rightarrow 1 - c_1 \cdot 1 - c_2 \cdot 1 - c_3 \cdot 2 - c_4 \cdot 2 - c_5 \cdot 1 = 0$$

We get $c_1 = 0, c_2 = 1, c_3 = 0, c_4 = 0, c_5 = 0$. Hence we can express $x_1^2x_2$ as $x_2^2 = x_2 \pmod{I(P)}$.

And we have now calculated our Gröbner basis:

$$(x_1^3 - x_1, x_1^2x_2 + x_1 - x_1x_2 - x_1^2, x_2^2 - x_2)$$

and

$$I(P) = (x_1^3 - x_1, x_1^2x_2 + x_1 - x_1x_2 - x_1^2, x_2^2 - x_2)$$

Example 11. *In this example we show that our elements in the Gröbner basis in the last example is in I(P). Again our points are $P = (p_1, p_2, p_3, p_4, p_5) = ((0,0,0,0,0), (1,0,0,0,0), (0,1,0,0,0), (1,1,0,0,0), (2,1,0,0,0))$ and our Gröbner basis is $(x_1^3 - x_1, x_1^2x_2 + x_1 - x_1x_2 - x_1^2, x_2^2 - x_2)$. For our first element we get:*

$$(x_1^3 - x_1)(p_1) = 0 - 0 = 0$$

$$(x_1^3 - x_1)(p_2) = 1 - 1 = 0$$

$$(x_1^3 - x_1)(p_3) = 0 - 0 = 0$$

$$(x_1^3 - x_1)(p_4) = 1 - 1 = 0$$

$$(x_1^3 - x_1)(p_5) = 2 - 2 = 0$$

and for the second element we get:

$$(x_1^2x_2 + x_1 - x_1x_2 - x_1^2)(p_1) = 0 + 0 - 0 - 0 = 0$$

$$(x_1^2x_2 + x_1 - x_1x_2 - x_1^2)(p_2) = 0 + 1 - 0 - 1 = 0$$
$$(x_1^2x_2 + x_1 - x_1x_2 - x_1^2)(p_3) = 0 + 0 - 0 - 0 = 0$$
$$(x_1^2x_2 + x_1 - x_1x_2 - x_1^2)(p_4) = 1 + 1 - 1 - 1 = 0$$
$$(x_1^2x_2 + x_1 - x_1x_2 - x_1^2)(p_5) = 1 + 2 - 2 - 1 = 0$$

And finally for the third:

$$(x_2^2 - x_2)(p_1) = 0 - 0 = 0$$
$$(x_2^2 - x_2)(p_2) = 0 - 0 = 0$$
$$(x_2^2 - x_2)(p_3) = 1 - 1 = 0$$
$$(x_2^2 - x_2)(p_4) = 1 - 1 = 0$$
$$(x_2^2 - x_2)(p_5) = 1 - 1 = 0$$

And as we can see all the elements in the Gröbner basis is in I(P).

4 Implementation

The program consists of a few functions that put together executes the algorithm. The code is about 800 lines including comments. The program uses the vector class from the c++ standard template library to handle the different points and vectors. And we have defined two types of containers using this class in this way:
typedef vector< vector<int> > Matrix;
typedef vector<int> Vector;

Now we take a closer look at some of the programs functions:

Vector pointMultiplyOrder(const Matrix &orderMatrix, const Matrix &pointMatrix, int dimension, int multElement);
This function takes a vector and multiplies it with our given order matrix to receive a new vector that can be compared with others lexicographically.

int adjustToField(int number, const int field);
This function takes an integer and if needed adjusts it to the field we're working in. By this we mean that if we are working in Z_p then the function makes sure that the integer sent in receives a new value between 0 and $(p-1)$. The function receives two variables; the integer number which is the input we wish to correct and the constant integer field which is the field we are working in.

Example 12. *adjustToField(15,7); means we want the number 15 to be adjusted to Z_7 and we receive the value 1.*

void sort(Matrix &matrix, const Matrix orderMatrix, const int numberOfPoints, const int dimension);
This function takes a set of vectors contained in the variable matrix and sorts them in the decided order who's matrix is contained in the variable orderMatrix.

void makeInversTable(Vector &inversTable, const int field);
This function finds the inverse for every element in the field and puts them in a table, or in this case a Vector.

Example 13. *makeInversTable(inverseVector,7) means we send in a Vector called inverseVector and the field we are working in, Z_7. The function will, to the variable inverseVector, return the following values 0,1,4,5,2,3,6 which are inverses of 0,1,2,3,4,5 and 6 respectively.*

void adjustVectorToField(Vector &row, const Vector inv, const int field, const int dimension);
This function finds the first nonzero element in a Vector and then uses the inverse table to find its inverse. Then it multiplies every element in the vector with that inverse. The purpose of this is to receive a vector who's first nonzero element is 1.

Example 14. *adjustVectorToField(vector,inverseTable,7,5); means that if we send in the vector (4,1,2,3,5) the function will multiply all the elements with the first elements inverse, so since the first element is 4 the inverse in Z_7 is 2 and vector receives the new values (1,2,4,6,3).*

void adjustMatrixToField(Matrix &matrix, const Matrix orderMatrix, const Vector inv, const int field, const int numberOfPoints, const int dimension);
This function takes every Vector in a Matrix and runs them through the function adjustVectorToField.

void Gauss(Matrix &matrix, const Matrix orderMatrix, Vector &inv, const int field, const int numberOfPoints, const int dimension);
This function uses Gauss-Jordan elimination to get a Matrix into reduced row echelon form.

bool isLinearlyIndependent(Matrix matrix, Vector testPoint, Vector inv, const int field, const int numberOfPoints, const int dimension);
This function compares a Vector to all the Vectors in a Matrix to see if the Vector is linearly independent or dependent to the Matrix.

void restOfL(Matrix &matrixL);
This function removes the first Vector from a Matrix.

Vector calculatePoint(Matrix pointMatrix, Vector exponentVector, const int field, const int numberOfPoints, const int dimension);
This function calculates the e(P) Vector that is used in the algorithm by multiplying the monomial e's exponent vector with the points given to the program.

Example 15. *calculatePoint(matrix,e,7,5,5) means if we send in a Matrix matrix with the points* $((1,0,0),(0,1,0),(1,1,0))$ *and the monomial* $e = x_2$ *then we get the output Vector (0,1,1).*

Matrix unionMatrixG(Matrix matrixG, Vector exponentVector);
This function takes a Matrix and a Vector and unites them into a new Matrix.

void deleteGfromL(Matrix &matrixL, Matrix matrixG, const int dimension);
This function deletes all multiples of monomials in one Matrix from another Matrix.

Matrix mergeL(Matrix matrixL, Matrix orderMatrix, Vector exponentVector, const int dimension);
This function merge two lists of monomials together.

Here is how the algorithm looks with the functions put together:

```
//Step 1
Matrix matrixL, matrixG, matrixB, baseMatrix;
Vector vectorE,vectorBase, temp(dimension);
int vectorsInBase=0;
matrixL.push_back(temp);
//step 2
while(vectorsInBase != numberOfPoints)
{
deleteGfromL(matrixL, matrixG, dimension);
vectorE = matrixL[matrixL.size()-1];
restOfL(matrixL);
//step 3 and 4
vectorBase = calculatePoint(points, vectorE, field, numberOfPoints, dimen-
sion);
bool independent = isLinearlyIndependent(matrixB, vectorBase, inv, field, vec-
torsInBase, vectorBase.size());
if(independent)
{
matrixB.push_back(vectorBase);
Gauss(matrixB, standardOrderMatrix2,inv,field,matrixB.size(),vectorBase.size());
baseMatrix.push_back(vectorE);
vectorsInBase++;
sort(matrixL, orderMatrix, matrixL.size(), dimension);
matrixL = mergeL(matrixL, orderMatrix, vectorE, dimension);
}
else
{
matrixG = unionMatrixG(matrixG, vectorE);
sort(matrixG, orderMatrix, matrixG.size(), dimension);
}
}
```

Table 1: Program efficiency results

Variables	Points	Seconds
5	10	1.4
5	15	22
5	20	170
5	25	836
10	10	1.6
10	15	22
10	20	174
10	25	821
20	10	7
20	15	28
20	20	182
20	25	830
30	10	30
30	15	61
30	20	209
30	25	878
40	10	110
40	15	161
40	20	334
40	25	1007

Running the program using different points and variables we get the results shown in the table. The results are from a laptop from 2010 with a Intel Atom 1.66 GHz processor. The points were chosen letting all variables except for the first two being zero and the first two variables being either the same or with one numbers difference going from zero and up. In all examples we worked in Z_{11}. The order used was the lexicographical order but DegLex and DegRevLex were also tested and both gave similar but slightly faster test results.
When comparing our programs efficiency with the efficiency of the program Macaulay 2 our program turns out to be a lot slower. The main reason for this is probably that in our program our order is defined by a matrix and therefore there are more calculations taking place than in Macaulay 2. Macaulay 2 can be downloaded for free from http://www.math.uiuc.edu/Macaulay2.

References

[1] J.A. Beachy, W. D. Blair, Abstact Algebra, third edition, Waveland Press, 2006.

[2] B. Buchberger and M. Möller, The construction of multivariate polynomials with preassigned zeroes. Computer algebra, Marseille, 1982.

[3] D. Cox, J. Little, D. O'Shea, Ideals, Varieties, and Algorithms, An introduction to computational Algebraic Geometry and Commutative Algebra, Springer, Second Edition, 1997.

[4] M.Kreuzer and L.Robbiano, Computational Commutative Algebra 1, Springer, 2008.

[5] R. Laubenbacher, B. Stigler, A computational algebra approach to the reverse engineering of gene regulatory networks. J. Theor 2004.

[6] S.Lundqvist, Complexity of Comparing Monomials and Two Improvements of the Buchberger-Möller Algorithm. MMICS 2008, Lecture Notes in Computer Science 5393 (2008) 105-125.

[7] L. Robbiano, Term orderings on the polynomial ring, EUROCAL'85, pages 513-517, 1985.