ALGEBRAIC RECONSTRUCTION OF GENE REGULATORY NETWORKS

Relatore interno:
Prof. Andrea Caranti

Relatore esterno: Dott. Giuseppe Jurman

Laureando: Andrea Gobbi

Anno accademico 2009-2010
## Contents

Introduction 5

1 Biological aspects 9
   1.0.1 Algebraic method 16

2 Basics of Gröbner basis 21
   2.1 Gröbner basis 21
   2.1.1 General facts 21
   2.1.2 Monomial order and multivariate division 22
   2.2 On 0-dimensional ideals 27

3 Discretization and interpolation 31
   3.1 Discretization 31
   3.1.1 SLC algorithm and a modification 32
   3.1.2 Linear discretization 34
   3.2 Interpolation over finite fields 35

4 Gröbner fan 39
   4.1 Universal Gröbner basis 39
   4.2 Polytope and general facts 40
   4.2.1 Polyhedral geometry and algebra 43
   4.3 Use of Gröbner fan for reverse engineering 44

5 Primary decomposition and vanishing ideal 47
   5.1 Vanishing ideal 47
   5.2 Primary decomposition 49
   5.2.1 Mathematical aspects 50
   5.2.2 Primary decomposition applied to reverse engineering 52
   5.3 Double primary decomposition 54
6 Stable border bases 57
  6.1 Mathematical aspects .................................. 57
  6.2 Use border bases for reverse engineering ............ 60

7 Linear codes 63
  7.1 Mathematical aspects .................................. 63
    7.1.1 Cyclic, Reed-Solomon and Reed-Muller codes .... 65
  7.2 Use codes for reverse engineering ..................... 66
    7.2.1 Possible improvements .............................. 67

8 Finite field method 69
  8.1 A new algorithm ........................................ 69

9 Test 73
  9.1 Gröbner fan and primary decomposition results ....... 76
    9.1.1 Comments ........................................... 81
    9.1.2 Conclusions ........................................ 85

A Appendix 87
  A.1 Groups, rings and ideals .............................. 87
  A.2 Polynomials ............................................ 89
  A.3 Finite fields .......................................... 92
Introduction

Even in the simpler organisms, no gene can be considered an independent entity within the genome: they are part of a number of highly intertwined groups of relations also involving their products, the proteins. Such sets of connections are nowadays known as (complex) networks, the gene regulatory network (GRN, for short) being probably the best known example: this entity collects the mutual (functional) relations linking one or more transcription functions to the affected genes and all the occurring feedbacks.

Reconstructing these relations has been a major task of molecular biology for the last decades, but only recently this process has become effective due both to theoretical and technical advances. The former is the collocation of the problem within the mathematical framework of graph theory: viewing the target network as a (directed and weighted) graph has opened the doors to a well built mathematical construction which has guaranteed a rigorous formalization of the objects and the methods. On the other side, the recently (from late nineties) developed high-throughput array techniques has supplied the biologists with a set of quick and relatively cheap tools for the quantitative simultaneous assessment of the expression levels of the involved players (DNA, mRNA, proteins). Both these factors have given a strong impulse to the aforementioned problem, so that nowadays the reconstruction of the edges of a biological networks starting from large-scale experiments is a key issue in computational biology, called reverse engineering. A typical case is the network reconstruction from a longitudinal experiment: the expression levels of all the genes (the graph nodes) is measured at different time steps of the biological process to be modeled, and the functional relations among them is investigated by means of their level variations across time.

The number of mathematical and statistical strategies designed to tackle the problem is constantly growing: they ranges from being purely probabilistic such as the Bayesian methods to totally deterministic approaches involving writing a dedicated differential equation for each node of the graph. A growing interest has recently arisen for deterministic approaches involving the use of algebraic methods representing the network as a polynomial dy-
namical system, first developed at Virginia Tech by the Laubenbacher Lab in 2003. The idea stems from the consideration that the microarray measurements are affected by a non negligible level of noise whose effect can be mitigated by quantizing them in a suitable finite field. Such discretization translates each involved function into a polynomial: within this new environment, the Gröbner basis and the primary decomposition of the polynomial ideals are the elective tools.

Although the results obtained through this techniques were encouraging on synthetic and biological networks, its pipeline includes a few ingredients which are indeed delicate, both from the theoretical and the computational point of view: choosing the best discretization field, correctly identifying the sought polynomials, taking care of the fundamental stochasticity of all biological process and so on. Concern about such points has been also confirmed in a few personal communication with the authors of the original papers on the topic. In this thesis we thoroughly analyze the proposed workflow highlighting the problematic points and proposing alternative improvements to each of them, providing both theoretical motivations and computational evidence. The introduced improvements come from very different branches of algebra, and the theory for some of them have been presented only very recently: this is the case of the stable border basis algorithm. It is worthwhile mentioning that network reconstruction (even regardless of the chosen algorithm) is a computationally hard task, and this thesis is no exception: computations play a key role here, and both dedicated computer algebra software and ad-hoc designed scripts have been employed for carrying on the required steps.

After a brief introduction to the biological background of gene regulatory network, the algebraic reconstruction method is described and a few points are discussed where problems may occur. For each of these points and the corresponding possible (hints of) solutions (discretization, Groebner fan, primary decomposition, stable border bases, linear codes and alternative finite field methods) a chapter is included, together with some background material on Gröbner basis. In each chapter, the underlying mathematics of the discussed topic is shown first, followed by examples of use for network reconstruction purposes. Finally, in the last chapter we demonstrate the advantages of the proposed techniques by applying them in a few experiments on a set of synthetic networks, generated by an ad-hoc simulator of biological regulatory networks and corresponding microarray time series. We conclude collecting the basics of the involved algebra in a dedicated Appendix.

As a final consideration, we observe that the original method may strongly benefit from the proposed improvements, but, as it happens in many reconstruction techniques, in terms of absolute performance on a real biological
example the availability of a priori knowledge on the functions (and thus the relations) of some of the involved genes is an invaluable piece of information which cannot be entirely substituted by a reconstruction algorithm, regardless of its complexity. On the other hand, starting from the same level of biological knowledge, the power of algebraic structures such as Gröbner basis (and, in general, polynomial ring theory) makes the presently considered algorithm one of the best available solution for the reverse engineering task.
Chapter 1

Biological aspects

The aim of this thesis is to propose some algebraic methods for the reconstruction of gene regulatory network. More specifically we want to illustrate some possible approaches for the reverse engineering of dynamic networks.

Definition 1. A gene regulatory network is the collection of molecular species and their interactions, which together control gene-product abundance.

This definition is taken from [KS09]. More specifically a gene regulatory network is a set of DNA segments inside a cell which interact with each other and with other substances in the cell. These interaction are indirect and entrusted to their RNA and protein expression products. The interactions govern the rates at which genes in the network are transcribed into mRNA. Each mRNA molecule goes on to make a specific protein (or set of proteins). This protein can be structural, and so accumulate at the cell-wall, or within the cell to give it particular structural properties. Some proteins serve only to activate other genes, and these are the transcription factors that are the main players in regulatory networks. Note that some transcription factors could be inhibitory. We can see this structure in figure 1.1. These networks have a fundamental role in every process of life e.g. cell differentiation, metabolism and signal transduction.

Definition 2. A graph is an ordered pair $G = (V, E)$ comprising a set $V$ of vertices or nodes together with a set $E$ of edges or lines, which are 2-element subsets of $V$.

A gene network can be seen as a graph $G$ in which the vertices $V$ are the genes and the edges $E$ are the interactions between them. We can figure out a network using four different types of graphs, depending on the information we have (or we need). The four types are summarized in figure 1.
In the networks shown in figure 1 the weights of the weighted graphs (or digraphs) are a measure of the strength of interactions. These weights and more generally the edges can be summarized in the adjacency matrix.

**Definition 3.** The adjacency matrix of a graph is a square matrix $n \times n$ such that every entry $a_{i,j}$ represents the relation between the two vertices $i$ and $j$ according to the type of network. We can see some of these matrices in table 1.1.

Note that the first two matrices are always antisymmetric and the last two are always symmetric.
**Definition 4. Gene expression** is the process by which information from a gene is used in the synthesis of a functional gene product. These products are often proteins, but in non-protein coding genes such as rRNA genes or tRNA genes, the product is a functional RNA. The process of gene expression is used by all known life - eukaryotes (including multicellular organisms), prokaryotes (bacteria and archaea) and viruses - to generate the macromolecular machinery for life. Several steps in the gene expression process may be modulated, including the transcription, RNA splicing, translation, and post-translational modification of a protein. Gene regulation gives the cell control over structure and function, and is the basis for cellular differentiation, morphogenesis and the versatility and adaptability of any organism. Gene regulation may also serve as a substrate for evolutionary change, since control of the timing, location, and amount of gene expression can have a profound effect on the functions (actions) of the gene in a cell or in a multicellular organism.

This definition is taken from [bio]. We are interested in the measurement of this gene expression, and, for the rest of the thesis, we will use "gene expression" as "measurement of gene expression".

A gene in the network can be influenced by some other genes in the network (also by itself) and the dependencies can be figured as the arc of the graph in figure 1. Every gene has a gene expression, and this measurement
depends on the interaction that the gene has in the network. If we know how
the network is, e.g. we know the topology and the quality of the dependencies,
we can recover the gene expression of all genes.

**Definition 5.** The reverse engineering is the process of developing a set
of specifications for a complex hardware system by an orderly examination
of specimens of that system. It is a process as being conducted by someone
other than the developer without the benefit of any of the original drawings
for the purpose of making a clone of the original hardware system.

This is a definition of the reverse engineering taken from [Rek85]. In other
words, reverse engineering is a sequence of operations that try to reconstruct
an original machine (a process, a system, a network) starting from different
outputs given by the original one when we propose different inputs.

In literature we can find a lot of approaches to this problem and we
summarize some of that (see [KS09]).

- The most basic and simplest model is the logic-based introduced by
  Kauffmann and Thomas. They use the idea that a gene can or not be
  active (1 and 0) and so they create a boolean network.

- Subsequently Shmulevich et al. modify the boolean approach introduc-
  ing a probabilistic component.

- Continuous model was introduced in order to avoid loss of informa-
  tion. The first method is a simple linear model, followed by ordinary
differential equations model. This last has the well-known problem in
  finding a solution.

- Other method are the Petri nets, single-molecule level and its approx-
  imation.

Make a reverse engineering in the contest of regulatory networks means
that we want to recover the network (the topology and, if possible, the quality
of the dependencies) starting from the gene expressions of all genes (that can
be measured in the laboratories).

In this predictive contest it is useful to used the so-called confusion
matrix. A confusion matrix, is a table like in table 1.2 with two rows and
two columns that reports the number \( a, b, c \) and \( d \), with:

- \( a \) is the number of correct predictions that an instance is negative,

- \( b \) is the number of incorrect predictions that an instance is positive,
• \(c\) is the number of incorrect predictions that an instance negative,

• \(d\) is the number of correct predictions that an instance is positive.

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Negative</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>Positive</td>
<td>b</td>
</tr>
<tr>
<td>Positive</td>
<td></td>
<td>c</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d</td>
</tr>
</tbody>
</table>

Table 1.2: Confusion matrix

The true positive rate (TP) is the proportion of positive cases that were correctly identified, as calculated using the equation:

\[ TP = \frac{d}{c + d}, \]

in the same way we can define the false positive rate (FP), the true negative rate (TN) and the false negative rate (FN) and they are calculated using the following equations:

\[ FP = \frac{b}{a + b} \quad TN = \frac{a}{a + b} \quad FN = \frac{c}{c + d}. \]

We can also introduce an accuracy indicator (AC) as:

\[ AC = \frac{a + d}{a + b + c + d}. \]

We can observe mutations in the gene expressions in an active network in different times, creating the so called timeseries.

Definition 6. A timeseries \(s\) is a \(m\)-upla of \(n\)-ulps such that for every time step \(1 \leq j \leq m\) the \(n\)-upla \(s_j\) has as components the gene expressions of all \(n\) genes. In other words, the element \(s_{j,i}\) is the expression at the time \(j\) of the gene \(g_i\).

We can figure a timeseries as a function \(\varphi : T \mapsto Q^n\), where \(T\) is the discrete ordered set of the time, and \(Q^n\) is the space in which the gene expressions live (here \(n\) stands for the number of genes in the network)\(^1\).

\(^1\)The choice of rational numbers \(Q\) instead of real numbers \(R\) is made because the gene expressions are laboratory measurements. From now the two fields will be used with the same meaning in this contest.
The reconstruction of a gene regulatory network consists in recovering the topology of the associated graph. The vertices are represented by genes and the edges are their interactions. From the timeseries of a network we want to recreate these links. According to the confusion matrix, we want to recover as much as possible the true positive and true negative edges and minimize the number of false positive and negative links.

The standard supposition is that the gene expression of a gene depends only by the expressions of all the genes at the previous temporary step. And all the algebraic methods that we will illustrate use this assumption. Note that such supposition implies that the timeseries is a Markov chain. Reconstruct the network starting form the gene expression means to find the relations between the genes. The expression of the gene $g_i$ is represented by a variable $x_i$. Our purpose is to find $n$ functions (transition functions) $f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots, x_n)$ such that $f_i(s_j) = s_{j+1,i}$, e.g. to find the $n$ functions that determinates the gene expressions in the time $j + 1$ knowing all gene expressions at the time $j$.

A timeseries consisting in 5 timesteps of three genes can be summarized in Table 1.3.

<table>
<thead>
<tr>
<th>Time</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.6104</td>
<td>1.2042</td>
<td>1.0072</td>
</tr>
<tr>
<td>2</td>
<td>1.7073</td>
<td>1.3252</td>
<td>1.0185</td>
</tr>
<tr>
<td>3</td>
<td>1.7254</td>
<td>1.4118</td>
<td>1.0336</td>
</tr>
<tr>
<td>4</td>
<td>1.7011</td>
<td>1.4616</td>
<td>1.0508</td>
</tr>
<tr>
<td>5</td>
<td>1.6601</td>
<td>1.4814</td>
<td>1.0685</td>
</tr>
</tbody>
</table>

Table 1.3: Typical timeseries

We will show now an historical example of a gene regulatroy network and its reconstruction.

**Example 1.** This is a simplified description of a regulatory network discovered by Jacob and Monod ([JM61]). This regulatory network earning them the 1965 Nobel Prize in Medicine. In prokaryotic organisms, some functions of regulation are accomplished by operons, consequential groups of genes that are transcribed as a single mRNA segment. Moreover operons have the control of the activation and inhibition (promoters, repressors) of the transcription of structural genes. The lac operon in *Escherichia coli* is one of such example. It enable *E. coli* to metabolize lactose into glucose in case glucose is not available directly. When glucose is present, the activity of the enzymes involved in the metabolism of lactose is very low, even if
lactose is available. In the other hand, when glucose is exhausted and lactose present, the concentration of lactose's enzymes increases. This process is called induction. We can summarize the situation in figure 1.3.

**Figure 1.3: The lac operon**

Before induction (higher part of figure 1.3) no lactose is present. The Active repressor can bind to the operator region of the lac operon and block RNA polymerase from transcribing the operon genes. After induction (lower part of figure 1.3) lactose and its isomer allolactose are present. The allolactose changes the conformation of the repressor protein which prevent it from blocking transcription. Consequently, RNA polymerase transcribes the operon into mRNA which, in turn, is translated into the proteins β-galactosidase, galactose permease and galactose acetylase. We will describe now how the lac operon is regulated. Positive controls takes place in the absence of glucose and presence of lactose. Some of the lactose is converted in allolactose, which, as we say before, acts as an inducer of transcription of the lac genes. This process involves the protein CAP (catabolite activator protein), which forms a complex with the substrate cAMP. Translation of the mRNA gives rise to the three proteins named before: galactose permease transports external lactose into the cell, β-galactosidase cleaves lactose into glucose and convert lactose into allolactose and galactose acetylase is not directly involved in lactose metabolism. Transcription continues until glucose becomes available. When that happens two negative control mechanism take
over. Synthesis of cAMP is inhibited and the repressor protein prevent the transcription \((\text{catabolite repression})\).

**Reconstruction**

The lac operon has been studied extensively and many different mathematical models have been constructed for it. The most common type of model is based on ordinary differential equations. The simplest model consists of three equations modelling the concentration of the repressor \(R\), the rate of change of the operon mRNA \(M\) and allolactose \(A\):

\[
\begin{align*}
R &= \frac{1}{1 + A^n} \\
M' &= c_0 + c(1 - R) - \gamma M \\
A' &= ML - \delta A - \frac{vMA}{h + A}
\end{align*}
\]

where \(R, M\) and \(A\) are functions of time \(t\) and the other letters are certain model parameters. This model is based on several assumptions (for more details see \([\text{LS07}]\)). One can compute the steady state of this system setting the right hand side of the differential equations to 0. In this case computer algebra allow us to describe the region in parameters space for which the dynamical system has more than one stationary point, and we can identify parameter values at which interesting phenomena might occur. Another interesting mathematical approach is in the form of Boolean network. In this case the variables can take value 0 or 1, corresponding to the biological assumption that the role of a molecular depends only on its absence or presence rather than a precise measure of its concentration. This model has nine boolean function and two parameters (external lactose and glucose). Also in this case computational algebra can help us to study the network. Indeed we can translate all boolean function into polynomials modulo 2. For more details see \([\text{LS07}]\).

1.0.1 Algebraic method

The first algebraic approach to the problem was made by R. Laubenbacher and B. Stigler in \([\text{LS04}]\). The basic idea is quite simple: all the functions in a finite field are polynomials. So we discretize the timeseries, we find a particular solution with a simple interpolation, and we reduce it by the ideal of the variety given by the time series. In order to do this we will use Gröbner basis. This important instrument can be used over finite fields and over infinite fields like \(\mathbb{R}\). The difference is that over \(\mathbb{R}\) we force the solution to be a polynomial and we may incur in the problem of overfitting. For this
reason we will use Gröbner basis over finite fields. Let us now exemplify the above workflow through a simple synthetic example.

**Example 2.** This is the timeseries we want to analyze. We will call the rows of the following table $s_1, \ldots, s_5$.

<table>
<thead>
<tr>
<th>Time</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.6104</td>
<td>1.2042</td>
<td>1.0072</td>
</tr>
<tr>
<td>2</td>
<td>1.7073</td>
<td>1.3252</td>
<td>1.0185</td>
</tr>
<tr>
<td>3</td>
<td>1.7254</td>
<td>1.4118</td>
<td>1.0336</td>
</tr>
<tr>
<td>4</td>
<td>1.7011</td>
<td>1.4616</td>
<td>1.0508</td>
</tr>
<tr>
<td>5</td>
<td>1.6601</td>
<td>1.4814</td>
<td>1.0685</td>
</tr>
</tbody>
</table>

We choose the field $\mathbb{F}_3$ and we project the points (discretization). The result is the following table:

<table>
<thead>
<tr>
<th>Time</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The next step is to find the three transition functions using some methods of interpolation:

\[
\bar{f}_1(x) = x_1^3 + x_1^2 x_3 - x_1^2 + x_1 x_3 - x_1 \\
\bar{f}_2(x) = x_1^3 - x_1^2 x_3 + x_1^2 - x_1 x_3 + x_1 + 1 \\
\bar{f}_3(x) = -x_1^3 - x_1^2 x_3 - x_1^2 - x_1 x_3 + x_1 + 1
\]

If $f$ is a solution, then $f + \mathcal{I}(\{s_1, \ldots, s_5\})$ is a solution. In this case:

\[
\mathcal{I}(\{s_1, \ldots, s_5\}) = \langle x_1 + x_2 - 1, x_2 x_3 - x_3^2 + x_2 - x_3, x_2^2 - x_3^2 + x_2 - x_3, x_1^3 - x_1, x_2^3 - x_2, x_3^3 - x_3 \rangle
\]

We fix now a monomial order $\prec$ and we calculate the Gröbner basis of the ideal above. We can reduce now the three functions above obtaining:

\[
f_1(x) = -x_3^2 + x_3 \\
f_2(x) = x_3^2 - x_3 + 1 \\
f_3(x) = -x_3^2 + x_2 + 1
\]

This means, in term of dependencies, that $f_1 = f_1(x_3)$, $f_2 = f_2(x_3)$ and $f_3 = f_3(x_2, x_3)$. Now we can draw the wiring diagram and the state space introduced by the functions above:

And finally we observe that there are a fixed point $\{(0,1,1)\}$ and three 3-cycle: $\{(0,1,0)\}, \{(0,1,2)\}$ and $\{(1,0,1)\}$. 
In this thesis we will analyze every point illustrated in the example above. The sketch in figure 1.6 summarizes the structure of this thesis:

(1) The problem of the discretization is discussed in chapter 3. We will show some methods that can be used to discretize the initial timeseries. Some of those have the level of the discretization as output (see [DML05]) others as input. This first step is important because it allows us to switch from $\mathbb{R}$ to $\mathbb{F}_p^n$. 
(2) In chapter 3 we will show two methods for the interpolation that are used to find a particular solution. This particular solution sends a timestep of the timeseries into the following one. We can not consider it as the solution to the problem because this is not unique.

(3) The solution obtained in the step 2 is not the only one possible, and so we will analyze the so-called Gröbner fan of an ideal. We have to consider all the possible solution and choose the more suitable (using some informations about the network, looking at the solutions with less variables possible or at the solution appears more frequently). These different solutions arise from different monomial orders that we can use to reduce the particular solution. This part is dealt specifically in chapter 4 and the theory is summarized in chapter 2.

(4) Finally we can recover all possible solutions using the point 2 and 3. In chapter 5 we can find an improvement of this approach and in chapters 6, 7 and 8 we will discuss some other possible methods. Finally in chapter 9 we will proposed some test and results.

Our work can be summarized as:

- A part in which we analyze some algebraic methods for reverse engineering. More specifically we focus the attention on the method that uses Gröbner basis.

- A part deal with the problems that are directly connected to the first step, e.g. the discretization and the interpolation.

- A part in which we propose some improvements, e.g. the double primary decomposition for the reduction of the possible models, the analysis of the weights distribution for the linear codes, a new method using the finite fields.

For our tests and for the computational part we use some computer algebra systems among which:

- *Sage* in the notebook version (see [S+10]),
- *Singular* (see [DGMGH10]) used inside *Sage*,
- *CoCoA System* (see [CoC]) using the graphic part and the server part,
- the *R*-package *netsim* (see [R D10] and [dCTC09]).
Chapter 2

Basics of Gröbner basis

In this chapter we will introduce some aspects about Gröbner basis and ideals. Set notation:

<table>
<thead>
<tr>
<th>Set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$</td>
<td>natural, integer, rational and real numbers</td>
</tr>
<tr>
<td>$\mathbb{E}$, $\mathbb{F}$, $\mathbb{K}$</td>
<td>fields</td>
</tr>
<tr>
<td>$\mathbb{A}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{K}$, \ldots</td>
<td>rings, groups or sets generally</td>
</tr>
<tr>
<td>$\mathbb{Z}_t$</td>
<td>integer modulo $t$</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^n}$</td>
<td>finite field with $p^n$ elements</td>
</tr>
</tbody>
</table>

2.1 Gröbner basis

When we have to decide if a vector $v$ is in a subspace $W$ of a vector space $V$ we use a basis of $W$. We know how to calculate it and, moreover, there is the idea of minimality. We would do the same with an ideal $\mathcal{I}$ in a polynomial ring $K[x_1, \ldots, x_n]$, e.g. when a polynomial $f \in K[x_1, \ldots, x_n]$ is an element of $\mathcal{I}$. Until the ’60 this problem seemed to be undecidable. Also using the Hilbert’s basis theorem, we can say only that $\mathcal{I}$ is finitely generated but the question of membership had no answer. In the 1965 Bruno Buchberger in his thesis developed the theory of the Gröbner basis. He named these objects after his advisor Wolfgang Gröbner. The problem shift from the minimal set (like the basis of vector spaces) to the quality of the set. In the following section we will describe the fundamentals of this theory and we give a more concrete description of the ideals.

2.1.1 General facts

In the appendix we can find the abstract definition of objects like rings, polynomials, polynomial rings and ideals. In the following lines we will describe...
more concretely what an ideal is. Let $K$ a field and let consider $K[x_1, \ldots, x_n]$ the polynomial ring in $n$ variables over $K$. An ideal $I$ of $K[x_1, \ldots, x_n]$ is a subset of $K[x_1, \ldots, x_n]$ that satisfies, according to the definition of ideal:

- $0 \in I$,
- if $f, g \in I$ then $f + g \in I$,
- if $f \in I$ and $g \in K[x_1, \ldots, x_n]$ then $fg \in I$.

Let $f_1, \ldots, f_m \in K[x_1, \ldots, x_n]$ and let

$$I = \left\{ \sum_{i=1}^m g_i f_i \text{ s.t. } g_i \in K[x_1, \ldots, x_n] \right\}.$$ 

It is easy to see that $I$ is an ideal of $K[x_1, \ldots, x_n]$. We can show also that every ideal of $K[x_1, \ldots, x_n]$ has this form for a right choice of $f_1, \ldots, f_m$. As a matter of notation, we shall indicate $I = \langle f_1, \ldots, f_m \rangle$. The question is: given $g \in K[x_1, \ldots, x_n]$, $g \in I$?

Let $n = 1$ and so $K[x_1, \ldots, x_n] = K[x]$. Let $I \subset K[x]$ an ideal, then exist a unique $h \in K[x]$ such that $I = \langle h \rangle$.

This characterization and the theorem 14 gives us a simple method to answer the previous question. Indeed we divided $g$ by $h$ and if the rest of the division is 0 then $g \in I$, otherwise $g \notin I$. We note that we can divide $g$ by $h$ only if the leading monomial of $g$ ($x^n$) and of $f$ ($x^m$) are such that $n \leq m$. This is an order over the terms.

We want to extend the notion of order over $K[x_1, \ldots, x_n]$ and then use a sort of Euclidean algorithm for answer to the previous question.

### 2.1.2 Monomial order and multivariate division

**Definition 7.** A total order is a binary relation (here denoted by infix $\prec$) on some set $S$. The relation is transitive, antisymmetric, and total.

Explicitly:

- if $a \prec b$ and $b \prec c$ then $a \prec c$
- if $a \prec b$ and $b \prec a$ then $a = b$
- $\forall a, b \in S \text{ or } a \prec b \text{ or } b \prec a$
A set paired with a total order is called a **totally ordered set**.

Let $K[x_1, \ldots, x_n]$ a polynomial ring over $K$ any field. We will denoted the monomials $x^n := x_1^{v_1} \cdots x_n^{v_n}$. We may consider $\alpha = (v_1, \ldots, v_n) \in \mathbb{N}^n$, in such way there is a bijection between the points of $\mathbb{N}^n$ and the primitive monomials of $K[x_1, \ldots, x_n]$.

**Definition 8.** A **term order** is a total order $\prec$ on $\mathbb{N}^n$ such that the 0 vector is the unique minimal element and $a \prec b$ implies $a + c \prec b + c$ for all $a, b, c \in \mathbb{N}^n$. (For more details see the appendix).

This last assertion says us that $\prec$ respects the product in $K[x_1, \ldots, x_n]$. Finally, every term order over $\mathbb{N}$ leads easily a monomial order over $K[x_1, \ldots, x_n]$. We say that $x^\alpha \prec x^\beta$ if and only if $\alpha \prec \beta$. For this reason we will use term order and monomial with the same meaning.

**Example 3.** Let $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ indicate the monomials $x^a$ and $x^b$. Moreover we indicate with $|a| = a_1 + \ldots + a_n$, e.g. the total degree of $x^a$. For the example we consider $R[x, y, z]$

- **Lexicographic** ($Lex$). We write $a \prec_{lex} b$ if $a_i < b_i$ where $i = \min \{ j \text{ s.t. } a_j \neq b_j \}$. Ex. $yz^2 \prec_{lex} xy^2 z$.

- **Graded lexicographic** ($DegLex$). We write $a \prec_{glex} b$ if $|a| < |b|$. In the case that $|a| = |b|$ we use $Lex$ for the comparison. Ex $xyz \prec_{glex} y^6$.

- **Weight order** There are two possibilities:

  1. Let $u = (u_1, \ldots, u_n) \in \mathbb{R}^n$ such that the coordinates are positive and linearly independent on $\mathbb{Q}$. Then we can give a term order using the dot product: we write $a \prec_u b$ if $u \cdot a < u \cdot b$. The linearly independence and the positivity assure $\prec_u$ to be a term order.

  2. We can also consider not a real vector but a vector $u \in \mathbb{N}^n$. In this case we have to fix also a monomial order $\prec$. In this way we can write: $a \prec_u b$ if $u \cdot a < u \cdot b$ or, if $u \cdot a = u \cdot b$, $a \prec b$. Note that $\prec$ is use to break ties.

This last weight order is a special case of the follow construction: we start with $u_1$ and to break ties we use an other $u_2 \in \mathbb{N}^n$. If there is still ties we introduce $u_3$ and so on. In [Rob86] we can found the proof that every term order arise in this way.

**Definition 9.** Given a term order, every non-zero polynomial $f \in K[x_1, \ldots, x_n]$ has a unique **leading monomial**, denoted $LM_{\prec}(f)$; the **leading coefficient** $LC_{\prec}(f)$ we is the coefficient of the monomial $LM_{\prec}(f)$.
We can use now a term order to reproduce Euclidean algorithm like in theorem 14. And we say that $f$ divides $g$, with $f, g \in K[x_1, \ldots, x_n]$ if and only if $LM_<(f)$ divides $LM_<(g)$ and this happens whenever that $LM_<(f) \prec LM_<(g)$.

Let $\mathcal{I} \subset K[x_1, \ldots, x_n]$ an ideal, then its leading ideal is the monomial ideal:

$$LM_<(\mathcal{I}) := \langle LM_<(f) : f \in \mathcal{I} \rangle$$

**Definition 10.** The monomials which do not lie in $LM_<(\mathcal{I})$ are called standard monomials and the set of the standard monomials, $N(\mathcal{I})$ is called $\Delta$-set or Gröbner escalier.

**Lemma 1.** The standard monomials form a $K$-vector space basis for the residue ring $K[x_1, \ldots, x_n]/\mathcal{I}$.

We will see later that an ideal with a finite number of elements in the $\Delta$ − set is a 0-dimensional ideal.

**Definition 11.** A finite set $\mathcal{G} \subset \mathcal{I}$ is a Gröbner basis for $\mathcal{I}$ with respect to $\prec$ if $LM_<(\mathcal{I})$ is generated by $\{LM_<(g) : g \in \mathcal{G}\}$. A minimal Gröbner basis for $\mathcal{I}$ is a Gröbner basis $\mathcal{G}$ of $\mathcal{I}$ such that:

- $LC_<(g) = 1 \ \forall \ g \in \mathcal{G}$
- $\forall \ g \in \mathcal{G} \ LM_<(g) \notin \langle LM_<(\mathcal{G} - \{g\}) \rangle$

It is called reduced if:

- $LC_<(g) = 1 \ \forall \ g \in \mathcal{G}$
- $\forall \ g \in \mathcal{G}, \text{ no monomial of } g \text{ lies in } \langle LM_<(\mathcal{G} - \{g\}) \rangle$

The reduce Gröbner basis is unique for an ideal and a term order. Starting with any set of generators of $\mathcal{I}$, the Buchberger algorithm compute the reduced Gröbner basis $\mathcal{G}$. The division algorithm rewrites each polynomial $f$ modulo $\mathcal{I}$ uniquely as a $K$-linear combination of standard monomials.

There are some important lemmas and theorems that justify all these assertion and for not overload the lecture they are all summarize below.

**Lemma 2.** Let $\mathcal{I} = \langle x^\alpha : \alpha \in A \rangle \subset K[x_1, \ldots, x_n]$. Then $h \in \mathcal{I}$ if and only if every monomial of $h$ is in $\mathcal{I}$.
2.1. GRÖBNER BASIS

Proof. $\Leftarrow$: Obvious.
$\Rightarrow$: If $h \in I$ then $h = \sum_{\alpha \in A} c_{\alpha} h_{\alpha} x^{\alpha}$ for some $h_{\alpha} \in K[x_1, \ldots, x_n]$ and $c_{\alpha} \in K$ with only a finite number of them not 0. Let $x^{\gamma}$ be a monomial of $h$. Obviously it is a monomial of at least one of $h_{\alpha} x^{\alpha}$ but if we write $h_{\alpha}$ as $h_{\alpha} = \sum_{i=1}^{m} \mu(i) x^{\beta(i)}$ then $x^{\alpha} h_{\alpha} = \sum_{i=1}^{m} \mu(i) x^{\beta(i)+\alpha}$.

Then exist $i$ such that $x^{\gamma} = x^{\beta(i)} x^{\alpha} \in I$. 

Lemma 3. In the same conditions of lemma 2, $x^{\gamma} \in I$ if and only in $x^{\gamma} = x^{\beta} x^{\alpha}$ with $\beta \in N^n$ and $\alpha \in A$.

Proof. The same of lemma 2

Lemma 4. Let $A \subset N^n$ and $I = \langle x^\alpha : \alpha \in A \rangle \subset K[x_1, \ldots, x_n]$. Then exists $A' \subset A$ finite such that $I = \langle x^\alpha : \alpha \in A' \rangle$.

Proof. This is the Dixon’s lemma. A proof can be found in [Dic13].

We can show directly now Hilbert’s theorem in this contest:

Theorem 1. Every ideal of $K[x_1, \ldots, x_n]$ is finitely generated.

Proof. Let $I \subset K[x_1, \ldots, x_n]$ an ideal and consider $J = \langle LM_{\prec}(I) \rangle$. For the lemma 4 $J$ is finitely generated, and so exist $g_1, \ldots, g_l \in I$ such that $J = \langle LM_{\prec}(g_i) | 1 \leq i \leq l \rangle$. These polynomials generated $I$. Indeed, let $f \in I$ by the division algorithm we find $h_1, \ldots, h_l, r \in K[x_1, \ldots, x_n]$ such that $f = \sum_{i=1}^{l} g_i h_i + r$ and no monomial of $r$ is divisible by any $LM_{\prec}(g_i)$. But $r = f - \sum_{i=1}^{l} g_i h_i$ and so $LM_{\prec}(r) \in J$ consequently by lemma 3 exist $i$ such that $LM_{\prec}(g_i) | LM_{\prec}(r)$. This last assertion say us that $r = 0$ and so $f = \sum_{i=1}^{l} g_i h_i$.

Theorem 2. For any ideal $I$ and a term order $\prec$ the Gröbner basis is finitely generated.

Proof. This is a consequence of theorem 1

We now explain the division algorithm and the Buchberger algorithm.

Algorithm 1. Let $f_1, \ldots, f_s, g \in K[x_1, \ldots, x_n]$ and $\prec$ a term order. This algorithm produce $h_1, \ldots, h_s, r$ such that $g = \sum_{i=1}^{s} h_i f_i + r$ and no monomial of $r$ is divisible by any $LM_{\prec}(f_i)$.

1. Put $\tilde{g} = g$ and $r = 0$. 

2. If \( \bar{g} = 0 \) the the algorithm ends.

3. Let \( x^\gamma = \text{LM}(\bar{g}) \), and \( c = \text{LC}(\bar{g}) \). If exist \( f_i \) such that \( x^\alpha = \text{LM}(f_i) \) divides \( x^\gamma \) then put

\[
\bar{g} = \bar{g} - \frac{c}{\text{LC}(f_i)} x^{\gamma-\alpha} f_i
\]

else put

\[
\bar{g} = \bar{g} - cx^\gamma \quad r = r + cx^\gamma.
\]

Then return to step 2.

It is easy to see that this algorithm ends and the requirements are satisfied.

Let \( G \subseteq K[x_1, \ldots, x_n] \) and \( f \in K[x_1, \ldots, x_n] \). We write \( r = \overline{f}^G \) if \( r \) is the rest of the division of \( f \) by \( G \). We say that \( f \) reduces to 0 modulo \( G \) if \( \overline{f}^G = 0 \). It is easy to see that if \( f \) reduces to 0 modulo \( G \), then reduces to 0 for any choice of the term order. Let \( f_1, f_2 \in K[x_1, \ldots, x_n] \setminus \{0\} \) and call \( x^{\alpha(i)} = \text{LM}(f_i) \). Let \( \gamma \) defines as \( \gamma_j = \max \{ \alpha(1)_j, \alpha(2)_j \} \). Then \( x^\gamma \) is called lowest common multiple (LCM) of \( x^{\alpha(1)} \) and \( x^{\alpha(2)} \). Moreover

\[
S(f_1, f_2) = \frac{x^\gamma}{\text{LC}(f_1)\text{LM}(f_1)} f_1 - \frac{x^\gamma}{\text{LC}(f_2)\text{LM}(f_2)} f_2
\]

is called S-polynomial of \( f_1 \) and \( f_2 \). Let \( f \in K[x_1, \ldots, x_n] \) and \( G \subseteq K[x_1, \ldots, x_n] \), if \( f \) reduces to 0 modulo \( G \), then \( f \) is a \( K[x_1, \ldots, x_n] \)-linear combination of element of \( G \).

Theorem 3. Let \( \mathcal{I} \subseteq K[x_1, \ldots, x_n] \) generated by \( G \). Then \( G \) is a Gröbner basis for \( \mathcal{I} \) if and only if \( S(g_1, g_2)^G = 0 \) for all \( g_1, g_2 \in G \).

Proof. The proof can be found in [CLO07], [Buc76]. \( \square \)

This theorem is useful for the creation of an algorithm that calculates a Gröbner basis of any ideal \( \mathcal{I} \) for any term order \( \prec \).

Algorithm 2. Let \( \{g_1, \ldots, g_s\} \) such that \( \mathcal{I} = \langle g_i : 1 \leq i \leq s \rangle \subseteq K[x_1, \ldots, x_n] \).

The algorithm calculate a Gröbner basis \( \mathcal{G} \) of \( \mathcal{I} \) respect to \( \prec \).

1. Put \( \mathcal{G}_0 = \{g_1, \ldots, g_s\} \) and \( i = 0 \).

2. If \( S(f, g)^{\mathcal{G}_i} = 0 \) for any \( f, g \in \mathcal{G}_i \), then \( \mathcal{G}_i \) is a Gröbner basis of \( \mathcal{I} \) and the algorithm ends.
3. If there exist $f, g \in G_i$ such that $\overline{S(f, g)} = r \neq 0$ then we put $G_{i+1} = G_i \cup \{r\}$ and $i = i+1$, then return to step 2.

The algorithm ends correctly.

**Example 4.** Let $\mathcal{I} = \langle f_1, f_2 \rangle = \langle x^3 - 2xy, x^2y - 2y^2 + x \rangle \subset \mathbb{Q}[x, y]$ with $grlex$ order. We have that:

- $\{f_1, f_2\}$ is not a Gröbner basis for $\mathcal{I}$ since $\overline{S(f_1, f_2)} = -x^2 \neq 0$.
- Using the last algorithm we obtain a Gröbner basis $\{f_1, f_2, f_3, f_4, f_5\}$ with $f_3 = -x^2, f_4 = -2xy, f_5 = -2y^2 + x$.
- In the construction of the basis we note that $f_1$ and $f_2$ can be eliminated, and multiplying the remaining three polynomial we have that $\{x^2, xy, y^2 - 1/2x\}$ is a minimal Gröbner basis.
- We note that also $\{x^2 + axy, xy, y^2 - 1/2x\}$ is a minimal Gröbner basis for any choice of $a \in \mathbb{Q}$.
- It is easy to see that the reduced Gröbner basis is the one with $a = 0$.

This is not an efficient algorithm. In the last years have been developed several algorithm for the calculation of a Gröbner basis of an ideal ([Fau99]). One of these use the fact that order term $DegRevLex$ leads the minimum computational time. Then if we want the Gröbner basis respect to any term order $\prec$ in the case that $\mathcal{I}$ is a 0-dimensional ideal, we start with the degrevlex basis and we applied $FGLM$ or Gröbner walk ([FGLM93], [FJLT07]).

### 2.2 On 0-dimensional ideals

Let $K$ a closed field. And let

\[
\left\{ \begin{array}{l}
p_1(x_1, \ldots, p_n) = 0 \\
\vdots \\
p_r(x_1, \ldots, p_n) = 0 
\end{array} \right. \tag{2.1}
\]

a polynomial system with $r$ polynomial equations in $n$ variables. The number of solution can be found looking at the ideal generated by $p_1, \ldots, p_n$.

**Definition 12.** Let $\mathcal{I} = \langle p_1, \ldots, p_r \rangle$, we define the **variety of an ideal** as:

\[
\mathcal{V}(\mathcal{I}) := \{ \bar{x} = (\bar{x}_1, \ldots, \bar{x}_n) \text{ s.t. } f(\bar{x}) = 0 \ \forall f \in \mathcal{I} \}
\]
Theorem 4. A solution of 2.1 exist if and only if $1 \notin \mathcal{I}$

Proof. This is the Hilbert’s Nullstellensatz theorem [Hil93]

We note that if $a = (a_1, \ldots, a_n) \in K^n$ is such that $p_1(a) = p_n(a) = 0$ then $f(a) = 0 \ \forall \ f \in \mathcal{I}$. This observation let us consider a Gröbner basis $\mathcal{G}$ for $\mathcal{I}$ instead of $p_1, \ldots, p_n$.

Theorem 5. The number of solution of 2.1 is finite if and only if the reduced Gröbner basis $\mathcal{G}$ for $\mathcal{I}$ is such that for all $i$ exist $g \in \mathcal{G}$ such that $LM(g) = x_i^h$ for some $h \geq 1$. And the number of solution is $|N(\mathcal{I})|$ (Definition 10).

Proof. The proof can be found in [CLO07].

\[ \begin{array}{c}
5 \\
4 \quad a_1 \quad b_3 \\
3 \quad \bullet \quad \mathcal{E}_1 \\
2 \quad \bullet \quad a_2 \quad \mathcal{E}_2 \\
1 \quad \bullet \quad \bullet \\
0 \quad \bullet \quad 1 \quad 2 \quad 3 \quad 4
\end{array} \]

Figure 2.1: A typical Gröbner escalier in two variables.

We shall do also viceversa.

Definition 13. Let $S = \{p_1, \ldots, p_s\}$ a set of points in $K^n$. We can define the vanishing ideal of $S$ as:

$$\mathcal{I}(S) = \{ f \in K[x_1, \ldots, x_n] : f(p_i) = 0 \ \forall p_i \in S\}.$$ 

Definition 14. A 0-dimensional ideal is an ideal $\mathcal{I}$ such that $0 \neq |\mathcal{V}(\mathcal{I})| < +\infty$.

Definition 15. The radical of an ideal of an ideal $\mathcal{I}$ is the ideal

$$\sqrt{\mathcal{I}} = \{ f \in K[x_1, \ldots, x_n] : f^m \in \mathcal{I} \ \text{for some} \ m > 0\}.$$ 

An ideal $\mathcal{I}$ is called radical ideal if $\mathcal{I} = \sqrt{\mathcal{I}}$.
If we have a set of points $S$, we may consider the vanishing ideal $\mathcal{I}(S)$, and then we shall return to the set considering the variety of the vanishing ideal $\mathcal{V}(\mathcal{I}(S))$. This last set is not generally the set $S$. In the same way we can start with an ideal $\mathcal{I}$, consider then the set $\mathcal{V}(\mathcal{I})$ and finally return to the ideal by $\mathcal{I}(\mathcal{V}(\mathcal{I}))$. Also in this case we generally do not obtain the initial ideal. We have that:

**Lemma 5.** Let $f_1, \ldots, f_m \in \mathbb{K}[x_1, \ldots, x_n]$, then $\langle f_1, \ldots, f_m \rangle \subset \mathcal{I}(\mathcal{V}(f_1, \ldots, f_n))$. We have the equality if and only if $\langle f_1, \ldots, f_m \rangle$ is radical. Vice versa, let $S \subset \mathbb{K}$, we have that $\mathcal{V}(\mathcal{I}(S)) \supset S$ and the equality holds if and only if $S$ is closed in $\mathbb{K}$.

**Algorithm 3.** Let $S = \{p_1, \ldots, p_s\}$ a set of points in $\mathbb{K}^n$, let $\prec$ any term order over $\mathbb{K}[x_1, \ldots, x_n]$. This algorithm returns the Gröbner basis of $\mathcal{I}(S)$.

1. Put $G_1 = \{x_1-p_{1,1}, \ldots, x_n-p_{1,n}\}$, where $p_{1,j}$ is the $j$-th coordinate of $p_1$.

2. If $k < s$ we have $G_k$ the basis for the first $k$ points we call $p_{k+1} = q$ and:

   - We arrange the $g_i \in G_k$ according to $\prec$ and $G_1 := \{g_1, \ldots, g_l\} \subset G_k$ such that $g_i(q) = 0$ for any $i \leq l$. If $G_1 = G_k$ then put $k = k + 1$ and go to step 3.
   - Let $g^*$ the first polynomial in $G_k$ such that $g^*(q) = \alpha^* \neq 0$. Define now $G_2 := \{g^* \cdot (x_1 - q_1), \ldots, g^* \cdot (x_n - q_n)\}$.
   - Define $G_3 := \\{g - \frac{g(p)}{\alpha^*} \cdot g^* \text{ for any } g > g^*\}$.
   - Set $G_{k+1} = G_1 \cup G_2 \cup G_3$ and $k = k + 1$

3. if $k = n$ the algorithm ends, otherwise return to step 2.

This is one of the algorithm improved to calculate the Gröbner basis of a vanishing ideal. The proof can be found in [FG06]. Others algorithms can be found in [MMM93], [ABKR00]. All these algorithm are based on the Buchberger-Möller algorithm [BM82].
Chapter 3
Discretization and interpolation

We say before that the result of the gene expression can be summarized as a matrix like table 1.3. Some algebraic method use to discretize the timeseries in order to use the finite field theory (for example the fact that every function is a polynomial). In the first section we will illustrate some results of the discretization theory and interpolation theory.

3.1 Discretization

A discretization of a real-valued vector \( v = (v_1, \ldots, v_n) \) is an integer-valued vector \( d = (d_1, \ldots, d_n) \) with the following properties:

- Each element of \( d \) is in the set \( \{0, 1, \ldots, D\} \) for some (usually small) positive integer \( D \), called the degree of the discretization.
- For all \( 1 \leq i, j \leq n \) we have \( d_i \leq d_j \) if \( v_i \leq v_j \).

This definition of discretization is taken form [Har01]. There are a lot of algorithms and techniques for the discretization: one common technique is based on clustering [JD88] and the most common algorithm is the k-means clustering developed by MacQueen [Mac67]. In [JD88] we can also find the SLC, single-link clustering. This algorithm is a divisive hierarchical clustering that defines the distance between two clusters as the minimal distance between two elements in two different clusters. The advantage to use this last technique is that at the beginning we need only the distance between points. The disadvantage of discretization is generally the loss of information. The SLC method requires \( D \) as input and this is potentially an other risk for the loss of information. The method described in [DML05] modifies SLC and implements in [Joh67] in way to avoid the two problem show above:
reduces the loss of information using Shannon’s entropy, and the degree of
the discretization is a output and not a input.

Let first give the definition of the Shannon’s entropy ([Sha48]). This article
was one of the founding works of the field of information theory. Suppose we
have a set of possible events whose probabilities of occurrence are \( p_1, \ldots, p_n \).
These probabilities are known but that is all we know concerning which event
will occur. The idea of Shannon was to find a function \( H(p_1, \ldots, p_n) \) that
measures how much choice is involved in the selection of the event. This
function, according to Shannon, has to satisfies this three properties:

- \( H \) should be continuous in the \( p_i \).
- If all the \( p_i \) are equal, \( p_i = \frac{1}{n} \), then \( H \) should be a monotonic increas-
ing function of \( n \). With equally likely events there is more choice, or
uncertainty, when there are more possible events.
- If a choice be broken down into two successive choices, the original \( H \)
should be the weighted sum of the individual values of \( H \).

Shannon showed that the only \( H \) satisfying these three assumption, is

\[
H = -K \sum_{i=1}^{n} p_i \log(p_i) \quad \text{with } K \text{ a positive constant.} \quad (3.1)
\]

### 3.1.1 SLC algorithm and a modification

The algorithm SLC uses graph theory. We start with a completed weighted
graph based on \( v \), this is the iteration 0. The following passage is to deleted
highest edges in order to disconnect the graph into two component. This is
the iteration 1. Then we consider every disconnected graph and we disconnect
again. The algorithm ends when the degree of the discretization is sufficient.
We disconnect a component if and only if both 1 and 2 below hold:

1. The component is not a complete graph. Or the minimum vertex
degree of the component is less than the number of its vertices
minus 1.

2. Any one of this three criteria holds:
   - The average edge weight of the component is greater than half of
   the average edge weight of the complete graph.
   - The distance between its largest and smallest vertices is greater
   than to half of this distance in the complete graph.
3.1. DISCRETIZATION

- If these two conditions fail, a third is applied: disconnect the component if it leads a substantial increase of Shannon’s entropy.

The first three criteria are like SLC algorithm. The last is used to measure the loss of information with Shannon’s entropy. Applying the equation 3.1 for our case we obtain:

\[ H = \sum_{i=0}^{n-1} \frac{w_i}{n} \log_2 \left( \frac{n}{w_i} \right). \]  

(3.2)

Where the base 2 of the logarithm is chosen so that the resulting units can be called bits, the number \( n \) is the number of states in which the vector is discretized and \( w_i \) is the number of entries discretized into state \( i \). We see that an increase of the number of states increase the entropy, with an upper-bound of \( \log_2 n \). We want that the number of states to be small. For this reason is useful to notice that \( H \) increase by different amount depending on which state is split and the number of elements in the new states. For example, if we split a large state in two equals parts the entropy increase much then if we split a little state in two different size states.

In the last criteria we consider ”substantial increase” if the component correspond to a large collection of entries. In our implementation a component will be disconnected only if contains al least half the vectors entries. And unlike the other three condition, when the fourth is applied, we split into 2 equals part (or with a difference of one unit if the number is odd), guaranteeing a maximum increase of the information measure.

**Algorithm 4.** With the previous notation, the discretized vector \( d \) can be reached with the following algorithm.

1. Construct a completed weighted graph \( G \) in which the vertex are distinct values of \( v \) and the weight of each edge is the Euclidean distance between the incident vertices.

2. Remove the edges of highest weight.

3. If \( G \) is disconnected into components \( C_1, \ldots, C_m \) then go to 4. Else, go to 2.

4. For each \( C_i \) apply the SLC criteria. If them hold then set \( G = C_i \) and go to 2, else go to 5.

5. Apply Shannon criteria. If it is satisfied, go to 6, else go to 7.
6. Sort the vertex values of $C_i$ and split them into two sets: if $|V(C_i)|$ is even, split the first half in a set and the second into another. If $|V(C_i)|$ is odd, split the first half +1 in a set, and the rest in another set.

7. Sort the components $C_i$, by the smallest vertex value in each $C_i$ and enumerate them $0, \ldots, D-1$ where $D$ is the number of component in which $G$ got disconnected.

For our purpose we need that the degree of the discretization is a number $p^n$ with $p$ a prime (little if possible). Suppose that a vector $v$ has been discretized into $m$ states as described in the previous algorithm. The first step is to find the smallest integer $k = p^n$ such that $m \leq k$. The discretization gives $m$ cluster, the remaining $k-m$ can be constructed by sorting the entries in each cluster and splitting the one that contains the two most distant entries. Then repeats it until $k$ cluster are obtained.

If we have more than a vector, suppose $N$, then we discretize them into $m_1, \ldots, m_N$ states, and set $m = \max\{m_i : i = 1, \ldots, N\}$. Now find the least possible $k = p^n \leq m$ and finally discretize the vectors in the same way in which we discretize a vector in a number required of states.

This type of discretization, that has the level of the discretization as output and uses the Shannon’s entropy, holds good results if we have one real value vector, or some real value vectors, each with the same level of discretization. Since the timeseries has different level of expressions, when we want to fix a level for all data, we loose a lot of information recovered before. For this reason, for our test, we use a more easy and light discretization, the linear ones. A completely different approach for the recovering of informations can be found in [CRT06].

### 3.1.2 Linear discretization

The simplest way to discretize a real value vector is the linear discretization.

**Algorithm 5.** Let $u = (u_1, \ldots, u_n) \in \mathbb{Q}^n$ be the input. The output $d = (d_1, \ldots, d_n) \in \mathbb{N}^n$ can be found following these steps:

1. Choose a prime $p$ as the level of discretization.\(^1\)

2. Standardize the vector $u$. This normalization follows 2 steps:

   - compute the mean of the vector $\bar{u} = \frac{\sum_{i=1}^{n} u_i}{n}$ and replace all $u_i$ with $u_i - \bar{u}$.

---

\(^1\)The choice of $p$ prime is only for our purpose, one can choose every entire greater than 1.
3.2. INTERPOLATION OVER FINITE FIELDS

- Divide each \( u_i \) by the standard deviation

\[
\sigma = \sqrt{\frac{\sum_{i=1}^{n}(u_i - \bar{u})^2}{n-1}} = \sqrt{\frac{\sum_{i=1}^{n} u_i^2}{n-1}},
\]

the last equality holds because the mean of the new \( u \) is set to 0.

3. Compute

\[
\Delta = \max_{i}\{u_i\} - \min_{i}\{u_i\}.
\]

4. Calculate every \( d_i := u_i \% \Delta \), where \( \% \) is the entire part of the division.
   If \( u_i = \max u \) put \( u_i = p - 1 \).

5. We have as output a entire vector \( d \) with components \( 0 \leq d_i \leq p - 1 \)

When we apply this method to the timeseries, we must be careful to discretize in the right way. Each gene has a different expression function and so we do not have to discretize a each row of th table 1.3 but each column, in other words, we have to discretize the vectors \((s_{1,i}, \ldots, s_{n,i})\) for each \( 1 \leq i \leq m \) and not the \( s_j \) for \( 1 \leq j \leq m \).

**Example 5.** We discretize table 1.3 using the level \( p = 3 \). In the following example involving table 1.3 we will use \( \mathbb{F}_3 = \{-1, 0, 1\} \) instead of \( \{0, 1, 2\} \). In term of dependencies this traslation does not yield any problem. The result is summarizes in table 3.1.

<table>
<thead>
<tr>
<th>Time</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.1: Discretization of table 1.3 with \( p = 3 \)

### 3.2 Interpolation over finite fields

There are several ways to interpolate a function over a finite field. We remark that a function over \( \mathbb{E} \) a finite field is a polynomial (see theorem 16). The **Lagrange interpolation** is the most common type of interpolation and we
CHAPTER 3. DISCRETIZATION AND INTERPOLATION

will show how it works. Let \((a_1, \ldots, a_k)\) distinct elements of \(E\) and \((b_1, \ldots, b_k)\) the image of \((a_1, \ldots, a_k)\) under a unknown function \(\psi\). We define for all \(i \leq k\)

\[
\delta_{a_i}(x) = \frac{(x - a_1) \cdots (x - a_{i-1}) \cdot (x - a_{i+1}) \cdots (a_i - a_k)}{(a_i - a_1) \cdots (a_i - a_{i-1}) \cdot (a_i - a_{i+1}) \cdots (a_i - a_k)}.
\]

This definition holds:

\[
\delta_{a_i}(x) = \begin{cases} 
1 & \text{if } x = a_i \\
0 & \text{if } x \neq a_i 
\end{cases}
\]

If \(\psi : E \rightarrow E\) is a function, then we can write it as:

\[
\vartheta(x) = \sum_{i=1}^{k} \delta_{a_i}(x)\psi(a_i).
\]

Note that \(\psi(a_i) = b_i\) are the initial data. If we have \(k = |E|\) we find a unique function \(\vartheta\) satisfying \(\vartheta(a_i) = b_i\). And so we put \(\psi(x) = \vartheta(x)\). If we have less than \(|E|\) points, we find a family \(\{\vartheta_s\}_{s \in S}\) of functions such that \(\vartheta_s(a_i) = b_i\) for all \(i \leq k\) and \(s \in S\). In particular the functions \(\vartheta_i\) are in the form \(\vartheta + \theta_i\) such that \(\theta(a_i) = 0\) for all \(i \leq k\). This kind of interpolation can be used also in \(R\) or \(Q\).

We now describe the interpolation used in [LS04]. Using the so-called Chinese Remainder Theorem ([Lan71]), we can give another algorithm for the interpolation over finite fields in the case that \(E = F_p\) for some prime \(p\). This algorithm is already modified for our purpose and so the notation could result not very clear.

Let \(s_1, \ldots, s_m\) a set of points in \((F_p)^n\) (a timeseries) and so we can write \(s_i = (s_{i,1}, \ldots, s_{i,n})\). We want to find, for all \(1 \leq i \leq n\), all polynomial \(f_i\) that send the state \(s_j\) into the \(i\)-th coordinate of the next state, i.e.

\[
f_i(s_j) = s_{j+1,i} \text{ for } j = 1, \ldots, m-1.
\]

The algorithm we will present soon give a particular solution \(\overline{f}_i\) and in the next section we will describe how to reach all solution. The input is a time series \(s_1, \ldots, s_m\) and the output the \(\overline{f}_1, \ldots, \overline{f}_n\).

**Algorithm 6.** This algorithm consists in three steps:

- For each \(1 \leq j \neq k < m\) such that \(s_j \neq s_k\) find the first coordinate \(l\) in which they differ and calculate

\[
b_{i,j}(x) = (s_{j,l} - s_{k,l})^{p-2}(x_l - s_{k,l})
\]

where \(x = (x_1, \ldots, x_n)\) and \(p\) is the number of elements of the finite field (as hypothesis).
3.2. INTERPOLATION OVER FINITE FIELDS

- Define
  \[ r_j(x) = \prod_{k=1}^{m-1} b_{j,k}(x). \]

- Finally, calculate a particular solution
  \[ \overline{f}_i(x) = \sum_{j=1}^{m-1} s_{j+1,i} r_j(x). \]

We note that \( r_j(s_j) = 1 \) and \( r_j(x) = 0 \) otherwise and this is sufficient to say that the algorithm works.

Using this last algorithm, when possible, or the Lagrange interpolation, we find a particular solution of the problem \( F(x) = (f_1(x), \ldots, f_n(x)) \).

**Example 6.** Using table 3.1 we apply the algorithm 6 and we obtain three polynomials like equations 3.3:

\[
\begin{align*}
\overline{f}_1(x) &= x_1^3 + x_1^2 x_3 - x_1^2 + x_1 x_3 - x_1 \\
\overline{f}_2(x) &= x_1^3 - x_1^2 x_3 + x_1^2 - x_1 x_3 + x_1 + 1 \\
\overline{f}_3(x) &= -x_1^3 - x_1^2 x_3 - x_1^2 - x_1 x_3 - x_1 + 1
\end{align*}
\]
CHAPTER 3. DISCRETIZATION AND INTERPOLATION
Chapter 4

Gröbner fan

In this chapter we will see the mathematical fundamental for the Gröbner fan of an ideal and its application to the reconstruction of the gene networks.

4.1 Universal Gröbner basis

Before discuss the polyhedral theory for the study of the Gröbner fan of an ideal, we show some useful results about Gröbner basis and ideals.

Theorem 6. Every ideal $I \subset K[x_1, \ldots, x_n]$ has only finitely many distinct leading ideals.

Proof. Suppose that $I$ has an infinite set $\Omega_0$ of distinct leading ideals. Let $0 \neq f \in I$. Since $f$ has only a finite number of terms and since each term lies in a element of $\Omega_0$, then exists a monomial $m_1$ in $f$ such that the set $\Omega_1 = \{M \in \Omega_0 : m_1 \in M\}$ is infinite. Since the monomial ideal $\langle m_1 \rangle$ is contained in a leading ideal of $I$, the lemma 1 tell us that the monomials outside $\langle m_1 \rangle$ are $K$-linearly dependent modulo $I$. Hence there exist a non-zero polynomial $g \in I$ such that none of its term lies in $\langle m_1 \rangle$. Like before, $g$ has a finite number of terms, and so exist $m_2$ in $g$ such that the set $\Omega_1 = \{M \in \Omega_1 : m_2 \in M\}$ is infinite. Iterating this construction we obtain an infinite strictly chain of monomials ideals:

$$\langle m_1 \rangle \subset \langle m_1, m_2 \rangle \subset \langle m_1, m_2, m_3 \rangle \subset \ldots$$

Since $K[x_1, \ldots, x_n]$ is Noetherian, this is a contradiction, and the proof ends.

\[\square\]

Definition 16. Let $I$ an ideal. A universal Gröbner basis $U$ for $I$ is a finite subset of $I$ such that $U$ is a Gröbner basis for $I$ respect to all term orders $\prec$ simultaneously.
Corollary 1. Every ideal \( \mathcal{I} \subset \mathbf{K}[x_1, \ldots, x_n] \) possesses a finite universal Gröbner basis \( \mathcal{U} \).

**Proof.** The proof descends immediately from theorem 6. \( \square \)

Let now indicate \( x = (x_1, \ldots, x_n) \) and \( \alpha = (a_1, \ldots, a_n) \).

**Definition 17.** Let \( w = (w_1, \ldots, w_n) \in \mathbf{R}^n \). For any polynomial \( f = \sum c_i x^{\alpha_i} \) we define the leading form \( LM_w(f) \) to be the sum of all terms \( c_i x^{\alpha_i} \) such that the inner product \( w \cdot \alpha_i \) is maximal. In the same way of the leading ideal we can define also:

\[
LM_w(\mathcal{I}) := \langle LM_w(f) : f \in \mathcal{I} \rangle
\]

This ideal is not necessarily a monomial ideal, however, it is whenever we choose \( w \) sufficiently generic.

Let \( w \) as before with all coordinates greater or equal than 0 and \( \prec \) be an arbitrary term order. We define, as in the example 3, a new term order \( \prec_w \) as follows: for \( \alpha, \beta \in \mathbb{N}^n \) we define

\[
\alpha \prec_w \beta \iff w\alpha < w\beta \text{ or } (w\alpha = w\beta \text{ and } \alpha \prec \beta)
\]

**Lemma 6.** Let \( \mathcal{I} \) an ideal, then we have that \( LM_{\prec}(LM_w(\mathcal{I})) = LM_{\prec_w}(\mathcal{I}) \).

**Definition 18.** If \( w \) is any real vector such that \( LM_{\prec}(LM_w(\mathcal{I})) = LM_{\prec}(\mathcal{I}) \) for some term \( \prec \), then we define \( w \) to be a term order for \( \mathcal{I} \). We also say that \( w \) represents \( \prec \) for \( \mathcal{I} \). And we define also the Gröbner region \( GR(\mathcal{I}) \) to be the set of all \( w \in \mathbf{R}^n \) such that \( LM_w = LM_{w'} \) for some \( w' \geq 0 \).

**Lemma 7.** Let \( \mathcal{I} \) an ideal. Let \( w, w' \in \mathbf{R}^n \) and \( \varepsilon > 0 \) sufficient small. Then:

\[
LM_w(LM_{w'}(\mathcal{I})) = LM_{w+\varepsilon w'}(\mathcal{I})
\]

### 4.2 Polytope and general facts

**Definition 19.** A polyhedron is a finite intersection of closed semi-spaces in \( \mathbf{R}^n \).

It is easy to see that a polyhedron \( P \) can be written as \( P = \{ x \in \mathbf{R}^n : Ax \leq b \} \) with \( A \) a matrix with \( n \) columns. If \( b = 0 \) then exist \( m \) vectors \( u_i \) in \( \mathbf{R}^n \) such that:

\[
P = pos(\{u_1, \ldots, u_m\}) = \{ \lambda_1 u_1 + \ldots + \lambda_m u_m : \lambda_i \geq 0 \} \quad (4.1)
\]
4.2. POLYTOPE AND GENERAL FACTS

**Definition 20.** A polyhedron like 4.1 is called **polyhedral cone** or simply **cone**. The polar of a cone $C$ is defined as:

$$C^* = \{ w \in \mathbb{R}^n : wc \leq 0 \ \forall \ c \in C \}$$

A polyhedron $Q$ which is bounded is called **polytope**.

Every polytope $Q$ can be written as a convex hull of a finite set of points, i.e.

$$Q = \text{conv}(\{v_1, \ldots, v_m\}) = \left\{ \sum_{i=1}^{m} \lambda_i x_i : \lambda_i \geq 0, \sum_{i=1}^{m} \lambda_i = 1 \right\} \quad (4.2)$$

**Definition 21.** Let $P$ a polyhedron and $w \in \mathbb{R}^n$. We define:

$$\text{face}_w(P) = \{ u \in P : wu \geq wv \ \forall v \in P \}$$

Every subset $F$ of $P$ with has this form is called **face** of $P$.

The relation "be a face of" among polyhedra is transitive, because:

$$\text{face}_{w'}(\text{face}_w(P)) = \text{face}_{w' + \varepsilon w}(P) \quad \text{for } \varepsilon > 0 \text{ sufficient small} \quad (4.3)$$

![Figure 4.1: Representation of transitive property "to be a face of"](image)

**Definition 22.** The **dimension** of a face $F$ is the dimension of its affine span. A face of codimension 1 is a **facet**. Face of dimension 0 and 1 are called **vertices** and **edges** respectively.

Every polytope is the convex hull of its vertices, and every cone is the positive hull of its edges. This means that representation 4.1 and 4.2 unique and minimal.

We can introduce a operation between two polyedra:
**Definition 23.** The **Minkowski addition** of polyhedra $P_1, P_2$ is defined in the natural sense:

$$P_1 + P_2 = \{ p_1 + p_2 : p_i \in P_i \}$$

It is easy to see that this sum is additive over the faces i.e. the face of the sum is the sum of the faces. This implies that a vertex $v$ of the sum $P_1 + P_2$ can be seen uniquely as the sum $p_1 + p_2$ with $p_i$ vertex of $P_i$. And so every vertex of the sum is a sum of vertices, the vice versa is not always true.

![Figure 4.2: Minkowski addition of two quadrangles](image)

**Theorem 7.** *Every polyhedron $P$ can be written as the sum $P = Q + C$ with a polytope $Q$ and a unique cone, recession cone, $C$.***

The proof can be found in [Sch86].

**Definition 24.** A **polyhedral complex** $\Delta$ is a finite collection of polyhedra such that:

1. If $P \in \Delta$ and $F$ a face of $P$, then $F \in \Delta$.
2. If $P_1, P_2 \in \Delta$ then $P_1 \cap P_2$ is a face of $P_1$ and $P_2$.

The **support** of a complex $\Delta$ is $|\Delta| = \bigcup_{P \in \Delta} P$. A complex $\Delta$ which consists of cones is called **fan**. A fan is **complete** if its support is $\mathbb{R}^n$. Let $P$ a polyhedron and $F$ a face of $P$, then we define the **normal cone** of $F$ at $P$ as: $\mathcal{N}_P(F) = \{ w \in \mathbb{R}^n : face_w(P) = F \}$ The **normal fan** $\mathcal{N}(P)$ of $P$ is the fan of normal cones $\mathcal{N}_P(F)$ where $F$ ranges over the faces of $P$.

Note that $\text{dim}(\mathcal{N}_P(F)) = n - \text{dim}(F)$. Moreover, if $F, F'$ are faces of $P$, then $F'$ is a face of $F$ if and only if $\mathcal{N}_P(F)$ is a face of $\mathcal{N}_P(F')$. The support of $\mathcal{N}(P)$ is the polar $C^*$ of the recession cone $C$ in the decomposition of $P$. We can see the cone $C^*$ as those functionals $w$ which give a bounded solution when maximized over $P$.

**Definition 25.** Two polytopes $Q_1, Q_2$ are called **strongly isomorphic** if $\mathcal{N}(Q_1) = \mathcal{N}(Q_2)$. 
4.2. POLYTOPE AND GENERAL FACTS

4.2.1 Polyhedral geometry and algebra

Let \( f = \sum_{i=1}^{m} c_i x^{e_i} \in K[x] = K[x_1, \ldots, x_n] \). The follow definition related polyhedral geometry to the computational algebra:

**Definition 26.** The **Newton polytope** associate to \( f \) is the polytope like in 4.2:

\[
\text{New}(f) = \text{conv}\{a_i : i = 1, \ldots, m\}
\]

The operation of multiplication between polynomials correspond to the Minkowski sum between the polyhedra:

**Lemma 8.**

\[
\text{New}(fg) = \text{New}(f) + \text{New}(g).
\]

**Proof.** It is sufficient show that the two polytopes have the same vertices. We note that

\[
\text{face}_w(\text{New}(fg)) = \text{New}(\text{LM}_w(f))
\]

also that

\[
\text{LM}_w(fg) = \text{LM}_w(f)\text{LM}_w(g) \quad \text{and} \quad \text{LM}_w(IJ) \supseteq \text{LM}_w(I)\text{LM}_w(J)
\]

These two consideration are sufficient to prove the theorem. The complete proof can be found in [Stu96].

We want to generalize the Newton polytope to the ideals. Let \( I \subset K[x_1, \ldots, x_n] \) an ideal.

**Definition 27.** Two weight vectors \( w, w' \in \mathbb{R}^n \) are called **equivalent** if and only if \( \text{LM}_w(I) = \text{LM}_{w'}(I) \).

**Theorem 8.** Each equivalence class of weight vectors is a relatively open convex polyhedral cone.

**Proof.** Let \( C[w] \) the equivalence class containing \( w \). Let \( \prec \) a term order and \( \mathcal{G} \) the reduced Gröbner basis of \( I \) with respect to \( \prec_w \). Then the follow formula holds:

\[
C[w] = \{w' \in \mathbb{R}^n : \text{LM}_{w'}(g) = \text{LM}_w(g) \ \forall g \in \mathcal{G}\}.
\]

Thanks to this formula we can see \( C[w] \) as an intersection of hyperplane and open half-space. For more details and for the complete proof see [Sch86].
The formula 4.4 has an important reformulation in terms of the reduced Gröbner basis $\mathcal{G}$:

$$C[w] = N_Q(\text{face}_w(Q)), \quad \text{where } Q = New\left(\prod_{g \in \mathcal{G}}\right) = \sum_{g \in \mathcal{G}} New(g).$$

Finally we can defined:

**Definition 28.** The Gröbner fan $GF(I)$ is the set of closed cones $C[w]$ for all $w \in \mathbb{R}^n$.

**Example 7.** In figure 4.3 we see the Gröbner fan of the ideal

$$\mathcal{I} = \langle x^2y - z, y^2z - x, z^2x - y \rangle \subset \mathbb{Q}[x, y, z]$$

and of the ideal

$$\mathcal{J} = \langle w^2 - x, x^2 - y, y^2 - z, z^2 - x \rangle \subset \mathbb{Q}[w, x, y, z]$$

![Figure 4.3: Gröbner fan of two ideals](image)

### 4.3 Use of Gröbner fan for reverse engineering

In section 3.2 we obtain a particular solution $F(x) = (\overline{f_1}(x), \ldots, \overline{f_n}(x))$ using the interpolation shown in the algorithm 6. Suppose now there are two polynomial functions $f_i, g_i \in E[x_1, \ldots, x_n]$ such that for each time step $j = 1, \ldots, m - 1$:

$$f_i(s_j) = s_{j+1,i} = g_i(s_j)$$
so that \((f_i - g_i)(s_j) = 0\) for all \(j\). This fact tell us that any two functions that interpolate the time series differ by a function \(\Psi\) identically equal to 0 on the given time series. For all the consideration made in section 2.2 the function \(\Psi\) lies in the vanishing ideal of the time series, i.e. let \(\mathcal{J} = \mathcal{I}(\{s_1, \ldots, s_m\}) = \{f \in S[x_1, \ldots, x_n] : f(s_i) = 0 \ 1 \leq i \leq m\}\) as the definition 13, then \(\Psi \in \mathcal{J}\). Finally, the solution can be found reducing the particular solution \(F(x) = (\overline{f_1}(x), \ldots, \overline{f_n}(x))\) by the ideal \(\mathcal{J}\). When we reduce we have to choose a term order \(\prec\) and do the euclidean algorithm in the multivariate version. The different choice of \(\prec\) leads a different reduction of the \(\overline{f_i}(x)\). Without extra knowledge of the network we cannot establish what is the more suitable term order for the reduction. We are more interested about the variables (genes) involved in the functions (relations) respect to the way in which they appear. For example \(f_1(x_1, x_2, x_3) = x_1 + x_2 x_3 + 1\) and \(g_1(x_1, x_2, x_3) = x_1 x_2 + 2 x_3^2\) say us that \(x_1\) depends from itself and from \(x_2\) and \(x_3\).

For this reasons we have to look to every possible term orders. The number of the genes in a network can be very large and so it is impossible to reduce the particular solution for all possible monomial order. In chapter 4 we say that the set of the term order can be partitioned using the equivalence relation in the definition 27. This say us that term orders in the same equivalence class give the same representation in term of polynomial, and so we look the Gröbner fan of \(\mathcal{J}\) and then we reduced the particular solution\(^1\). Finally, we take the relationships that more frequently appears in the reduction of the particular solution.

**Example 8.** According to the previous consideration we have to calculate the vanishing ideal of the points in table 3.1 (a reduced Gröbner basis) \(\mathcal{J}\) using the Buchberger algorithm. Let \(\mathcal{J} = \langle g_1, \ldots, g_l \rangle\) we have to add the field equations, in this case \(\{x_1^3 - x_1\}\), this because we are in \(\mathbb{F}_3^2\). We have that \(\mathcal{J}\) is generated by:

\[
\langle x_1 + x_2 - 1, x_2 x_3 - x_3^2 + x_2 - x_3, x_2 x_3 - x_3^2 + x_2 - x_3, x_1^3 - x_1, x_2^3 - x_2, x_3^3 - x_3 \rangle
\]

(4.5)

Now we reduce the particular solution 3.3 by 4.5 obtaining:

\[
\begin{align*}
  f_1(x) & = -x_3^2 + x_3 \\
  f_2(x) & = x_3^2 - x_3 + 1 \\
  f_3(x) & = -x_3^2 + x_2 + 1
\end{align*}
\]

(4.6)

\(^1\)The cones of the Gröbner fan are in bijection with the reduced Gröbner basis of the ideal considered.

\(^2\)In *Singular* we use the function `vanishId` and we have to declare before the ring used and we do not need to add the field equation.
The problem is that this is not the unique reduction of the particular solution. We have to look at the Gröbner fan of $\mathcal{J}$. We calculate the Gröbner fan of the ideal and the function\(^3\) gives us the list of the weight vector which represent each equivalence class:

\[(3, 2, 1) \quad (2, 1, 2) \quad (2, 3, 1) \quad (1, 2, 2)\]

These weight vector are the output of the function $\text{groebner\_fan}(\text{ideal}).\text{weight()}$ and they define a term order using $\text{DegRevLex}$ for break ties.

We compute all possible reduction using the suitable term order. The reductions are summarized in table 4.1.

<table>
<thead>
<tr>
<th></th>
<th>$f_1(x)$</th>
<th>$f_2(x)$</th>
<th>$f_3(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$-x_3^2 + x_3$</td>
<td>$x_3^2 - x_3 + 1$</td>
<td>$x_2 - x_3 + 1$</td>
</tr>
<tr>
<td></td>
<td>$-x_2^2 - x_3 - x_2$</td>
<td>$x_2^2 + x_3 + x_2 + 1$</td>
<td>$-x_3^2 + x_3 + 1$</td>
</tr>
<tr>
<td></td>
<td>$-x_3^2 + x_3$</td>
<td>$x_2^3 - x_3 + 1$</td>
<td>$-x_3^2 + x_3 + 1$</td>
</tr>
<tr>
<td></td>
<td>$-x_1^2 - x_3 + 1$</td>
<td>$x_1^2 + x_3$</td>
<td>$-x_1^2 + x_3 - x_1$</td>
</tr>
</tbody>
</table>

Table 4.1: Possible reduction using Gröbner fan.

\(^3\)We used the function $\text{groebner\_fan}$ implemented in Sage using the program $\text{gfan}$ [Jen].
Chapter 5

Primary decomposition and vanishing ideal

In order to avoid long computational time and a lot of possible reduced polynomials, we try to pre-process the timeseries. In the first section of this chapter we will illustrate the basically idea that we developed our own and then we find a justification and a mathematical explanation in [JLSS07]. In the last section we will present a possible improvement of the idea.

5.1 Vanishing ideal

Let $s_1, \ldots, s_m$ the time series to analyze. Instead of search for the $\bar{f}_i(x)$ and then reduce, we can modify the time series as follow:

**Algorithm 7.** This was the idea until we discover [JLSS07].

- Set $p^i_j = (s_j | s_{j+1,i}$ for all $1 \leq j \leq m - 1$ and $1 \leq i \leq n$. This means that we append to the $s_j$ the $i$-th coordinate of the following time step $s_{j+1}$

- Using the Buchberger algorithm 3 for all $1 \leq i \leq n$ we find the universal Gröbner basis of the vanishing ideal $I_i = I(\{p^i_1, \ldots, p^i_{m-1}\})$ of these points, using for example the algorithm shown in [BOT02].

- Watching at these ideals we can find some possible solution. Indeed we isolate the last variable in the generators of $I_i$ if it is possible and in this way we know which variables are involved.

In the previous section we have that $\bar{f}_i(s_j) = s_{j+1,i}$ and this means that $\bar{f}_i(s_j) - s_{j+1,i} = 0$. We can consider this last relation as a $g \in S[x_1, \ldots, x_n, h]$
such that \( g(s_j, s_{j+1,i}) = 0 \), and so consider the points \( p_j^i = (s_j, s_{j+1,i}) \)  
and the vanishing ideal in which \( g \) lies. In \( I_i \) we have some polynomials in \( x_1, \ldots, x_n, h \). If in these polynomials the variable \( h \) does not appear, we discard them, otherwise we try to isolate it. The interesting case is when a polynomial is linear in \( h \), e.g. \( h + f(x_1, \ldots, x_n) \). It is quite obvious that starting from this form we can recover the \( \mathcal{I}_i \).

Also in this case we have to choose between the representative of the ideal. We follows show how this algorithm works. Note that we do not calculate the universal Gröbner because we realize after that there was a more interesting approach using the primary decomposition.

**Example 9.** Using table 3.1 we construct the required points as we can see in table 5.1.

<table>
<thead>
<tr>
<th></th>
<th>( g_1 )</th>
<th>( g_2 )</th>
<th>( g_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(2,2,2,1)</td>
<td>(2,2,2,0)</td>
<td>(2,2,2,2)</td>
</tr>
<tr>
<td>2</td>
<td>(1,0,2,1)</td>
<td>(1,0,2,0)</td>
<td>(1,0,2,0)</td>
</tr>
<tr>
<td>3</td>
<td>(1,0,0,0)</td>
<td>(1,0,0,1)</td>
<td>(1,0,0,1)</td>
</tr>
<tr>
<td>4</td>
<td>(0,1,1,0)</td>
<td>(0,1,1,1)</td>
<td>(0,1,1,1)</td>
</tr>
</tbody>
</table>

Table 5.1: Pre-processing of table 3.1

We calculate the vanishing ideal using \( \text{DegRevLex} \) for \( g_1 \) and we obtain the following reduced Gröbner basis:

\[
\mathcal{I}_1 = \langle x_1 + x_2 - 1, h^2 - h, x_3 h + h, x_2 h - x_2 + x_3 + h, x_3 - x_3 + h, x_2 x_3 + x_2 + x_3 + h, x_2^2 + x_2 + x_3 + h \rangle
\]

We are interested of the last 3 polynomials:

\[
x_3^2 - x_3 + h, x_2 x_3 + x_2 + x_3 + h, x_2^2 + x_2 + x_3 + h.
\]

Isolating \( h \) we obtain:

\[
h = -x_3^2 + x_3, \tag{5.1}
\]

\[
h = -x_2 x_3 - x_2 - x_3,
\]

\[
h = -x_2^2 - x_2 - x_3.
\]

In the same way, for the gene \( g_2 \) we obtain:

\[
\mathcal{I}_2 = \langle x_1 + x_2 - 1, h^2 - h, x_3 h - x_3 + h - 1, x_3^2 - x_3 - h + 1, x_3 h - x_3 + h - 1, x_2 x_3 + x_2 + x_3 - h + 1, x_2^2 + x_2 + x_3 - h + 1 \rangle
\]
5.2. PRIMARY DECOMPOSITION

Isolating \( h \) we obtain:

\[
\begin{align*}
  h &= x_2^2 + x_2 + x_3 + 1, \\
  h &= x_2x_3 + x_2 + x_3 + 1, \\
  h &= x_3^2 - x_3 + 1.
\end{align*}
\]

(5.2)

Finally for \( g_3 \)

\[
\begin{align*}
  \mathcal{I}_3 &= \langle x_1 + x_2 - 1, h^2 + x_2 - x_3 - 1, x_3h - x_3 + h - 1, x_3^2 - x_2 + h - 1, \\
  &\quad x_2h - x_3 + h - 1, x_2x_3 - x_3 + h - 1, x_2^2 - x_3 + h - 1. \rangle
\end{align*}
\]

Isolating \( h \) we obtain:

\[
\begin{align*}
  h &= -x_3^2 + x_2 + 1, \\
  h &= -x_2x_3 + x_3 + 1, \\
  h &= -x_2^2 + x_3 + 1.
\end{align*}
\]

(5.3)

Comparing these results with example 8 we note that some polynomial are similar.

The approach shown in the algorithm 7 is a good strategy but somehow informal. In the following sections we will see some facts and theorems that give us a formal method using the primary decomposition.

5.2 Primary decomposition

We want to emphasize that a lot of variables occurring into the polynomials are redundant. The Gröbner fan approach does not look at this problem. With a little pre-processing data we can reduce the number of variables or the number of possible models describing the problem e.g. for all gene \( g_i \), which gene expression is the variable \( x_i \), we can find all sets of variables \( x_{i_1}, \ldots, x_{i_s} \) such that \( f_i(x_1, \ldots, x_n) = f_i(x_{i_1}, \ldots, x_{i_s}) \) and this sets are minimal\(^1\). The method is based on the primary decomposition and does not depend on the choice of a term order. The algorithm can be found in [JLSS07].

We fix a gene \( g_i \) and we consider the timeseries like in subsection 5.1 e.g. in the form \((s_k, t_k)\). It is obvious that \( s_k \in (\mathbb{F}_q)^n \) and \( t_k \in \mathbb{F}_q \). For \( a \in \mathbb{F}_q \) set:

\[
X_a = \{ s_i \text{ s.t. } t_i = a \},
\]

\(^1\)Minimal means that this polynomial does not exist if we one variable is removed.
and let \( X = \{ X_a \}_{a \in \mathbb{F}_q} \) a partition of the timeseries respect to the last variable. Let now \( Y = \{ f \in \mathbb{F}_q[x_1, \ldots, x_n] \text{ s.t. } f(p) = a, \forall p \in X_a, a \in \mathbb{F}_q \} \).

The elements of \( Y \) are all the possible polynomials we are interested for. We want to find the polynomial(s) with a minimal number of variable. This is a natural choice. It is easy to increase the number of variables (adding the field equations for example), but is quite difficult to reduce it. Before to proceed with the method, we need some instruments.

### 5.2.1 Mathematical aspects

**Definition 29.** An ideal \( I \subset K[x_1, \ldots, x_n] \) is called **primary** if \( fg \in I \) implies \( f \in I \) or some power \( g^m \in I \) for some \( m > 0 \).

**Lemma 9.** Every ideal \( I \) can be written as a finite intersection of primary ideals.

**Definition 30.** A **primary decomposition** of an ideal \( I \) is an expression of \( I \) as an intersection of primary ideals \( I = \bigcap_{i=1}^{s} Q_i \). It is called **minimal** if the \( \sqrt{Q_i} \) are all distinct and \( Q_i \nsubseteq \bigcap_{j \neq i} Q_j \).

**Theorem 9.** Every ideal \( I \) has a minimal primary decomposition.

This last theorem is the know Lasker-Noether theorem, and the proof can be found in [CLO07] together to the proof of lemma 9.

**Definition 31.** A **simplex** is the generalization of a tetrahedral region of space to \( n \) dimensions. It is the simplest possible polytope in \( n \)-space. A simplex in \( n \)-space is regular if and only if all its edges are the same length. All the lower-dimensional elements of a simplex are themselves simplex.

A curious fact: the number of \( k \)-dimensional elements of an \( n \)-dimensional simplex is given by the coefficient of \( a^{k+1} \) in the binomial expansion of \((a + 1)^{n+1}\). For example, the binomial expansion of \((a + 1)^8\) is:

\[
a^8 + 8a^7 + 28a^6 + 56a^5 + 70a^4 + 56a^3 + 28a^2 + 8a + 1.
\]

From this, a seven-dimensional simplex has:

- one hepton (itself, an octahexon),
- eight hexons (heptapenta: hence the name octahexon),
5.2. PRIMARY DECOMPOSITION

- 28 pentons (hexatetra),
- 56 tetrons (pentachora),
- 70 cells (tetrahedra),
- 56 faces (triangles),
- 28 edges,
- eight vertices
- one nullitope.

Definition 32. A simplicial complex $\Delta$ is a set of simplices such that:

- Any face of a simplex from $\Delta$ is also in $\Delta$.
- The intersection of any two simplices $\sigma_1, \sigma_2 \in \Delta$ is a face of both $\sigma_1, \sigma_2$.

Note that the definition 32 is very similar to definition 24.

Definition 33. For $F \subset \{1, \ldots, n\}$, let $R_F = F_q[x_i \text{ s.t. } i \notin F]$. Let

$$\Delta_X = \{ F \text{ s.t. } Y \cap R_F \neq \emptyset \}.$$ 

It is easy to see that $\Delta_X$ is a simplicial complex.

Definition 34. The Alexander dual of a simplicial complex $\Delta$ with vertices $[n]$ is the simplicial complex defined as:

$$\Delta^* = \{ [n] \setminus F \text{ s.t. } F \notin \Delta \}$$

Definition 35. In the situation above, we define a square-free monomial ideal $M_X \subset F_q[x_1, \ldots, x_n]$ generated by:

$$\{ m(p, q) \text{ s.t. } p \in X_a, q \in X_b, \text{ and } a \neq b \in F_q \}$$

where

$$m(p, q) = \prod_{p_i \neq q_i} x_i.$$ 

Note that $m(p, q)$ is a monomial which encode where $p$ differs from $q$. Moreover $M_X$ is the same as the face ideal for Alexander dual of $\Delta_X$. For this reason holds the following lemma:
Lemma 10. Let $F \subset \{1, \ldots, n\}$ then $F \in \Delta_X$ if and only if the ideal $\langle x_i \text{ s.t. } i \notin F \rangle$ contains the monomial ideal $M_x$.

The proof can be found in [JLSS07]. An immediate corollary is:

Corollary 2. The minimal subsets $F$ such that $Y \cap R_F \neq \emptyset$ are exactly the generating sets for the minimal primes in the primary decomposition of the ideal $M_X$.

5.2.2 Primary decomposition applied to reverse engineering

We are ready now to write the algorithm that finds all minimal sets:

Algorithm 8. Using the previous notation the algorithm is divided into three steps:

1. Compute the monomial ideal $M_X$.
2. Compute the primary decomposition of $M_X$, and finally
3. compute the generating sets of all minimal primes of $M_X$.

Example 10. We use the timeseries in table 5.1 and we look only at the gene $g_1$. We have to write the time series in the form $(s_k, t_k)$. In this case $t_k = s_{k+1,1}$ because we are studying $g_1$. We have:

\[
\begin{align*}
(s_1, t_1) &= ((2, 2, 2), 1) \\
(s_2, t_2) &= ((1, 0, 2), 1) \\
(s_3, t_3) &= ((1, 0, 0), 0) \\
(s_4, t_4) &= ((0, 1, 1), 0)
\end{align*}
\]

Then $X_0 = \{(2,2,2),(1,0,2)\}$, $X_1 = \{(1,0,0),(0,1,1)\}$ and $X_2 = \emptyset$. In this situation we have to calculate $m(p, q)$, as the definition, with $p \in X_0$ and $q \in X_1$. And so $m((2,2,2),(1,0,0)) = x_1x_2x_3$ since they differ in every coordinate. In the same way we have $m((2,2,2),(0,1,1)) = x_1x_2x_3$, $m((1,0,2),(1,0,0)) = x_3$ and finally $((1,0,2),(0,1,1)) = x_1x_2x_3$. The ideal $M_X = \langle x_1x_2x_3, x_3 \rangle = \langle x_3 \rangle$. In this case the primary decomposition and the generating set is obvious only $\{x_3\}$. This means that $f_1(x_1, x_2, x_3) = f_1(x_3)$ according to the previous results.
5.2. PRIMARY DECOMPOSITION

This algorithm can be implemented in *Singular* using the function *primdecGTZ* in the library *primdec.lib*.

We want to establish which is the most relevant minimal sets, this because the number of the possible decomposition can be very large. For do this we can use information about the topology (generally not available) or make some statistical indicators. We will show some of them taken from [JLSS07].

The first step is to look at the length of the decompositions. Let

\[ F = \{ F_1, \ldots, F_l \} \]

the set of the generating sets of all minimal primes of \( M_X \) e.g. the output of the algorithm above.

For \( 1 \leq s \leq n \), let \( Z_s \) be the number of sets \( F_j \in F \) of lengths, and let \( W_i(s) \) be the number of sets \( F_j \in F \) of lengths and such that \( x_i \in F_j \). Jarrah et al in [JLSS07] propose three different methods to score each variable \( x_i \). Let

\[
S_1(x_i) = \sum_{s=1}^{n} \frac{W_i(s)}{sZ_s},
\]

\[
S_2(x_i) = \sum_{s=1}^{n} \frac{W_i(s)}{s},
\]

\[
S_3(x_i) = \sum_{s=1}^{n} W_i(s).
\]

These indicators are used to score every minimal set \( F_j \in F \):

\[
T_1(F_j) = \prod_{x_i \in F_j} S(x_i),
\]

\[
T_2(F_j) = \frac{\sum_{x_i \in F_j} S(x_i)}{|F_j|}.
\]

We have now a set of possible models e.g. a subset of coordinates that describe entirely the selected gene. We have to repeat this pre-process for all gene, in this way every polynomial has been to search in a subset of the set of all coordinates. After this pre-processing we can use the method using Gröbner fan described in subsection 4.3 or the other methods remembering that we can remove some coordinate according to the pre-processing algorithm. In this way we reduce the possibilities of the polynomials and the computational time.
5.3 Double primary decomposition

The primary decomposition used before allow us to know the dependencies without calculate directly the functions $f_i$. This preprocessing is useful when we have a lot of gene and timestep. After the preprocessing we can find the functions using only the variables we need, in other words, if $\{x_{i_1}, \ldots, x_{i_m}\} \subset \{x_1, \ldots, x_n\}$ is a minimal subset describing the solution, we can consider a sub-timeseries with the values only in $i_1, \ldots, i_m$ and so reduce the computational time. Experimentally we observe that the set of minimal subsets can be very large. For this reason we introduce a new preprocessing algorithm. Using the primary decomposition, we developed a new method that is the natural consequence of the algorithm 8. The first step is to look the timeseries "horizontally", in other words to look if there are dependencies between the genes at the same temporally step. We will call it intrinsically dependencies.

Algorithm 9. Initially, before learn about primary decomposition, the algorithm was:

- consider every time step as a point of $(F_q)^n$.
- for every gene $i$ we
  1. set a lexicographic weighted order $\prec$ such that $x_i > x_j$ for any $j \neq i$.
  2. use the Möller algorithm and find the reduced Gröbner basis $B$.
  3. look if in $B$ there is a polynomial of degree 1, e.g. a polynomial in the form $x_i - f(x_J)$ with $J \subset \{1, \ldots, n\} \setminus \{i\}$.
  4. this say us that at every level the expression of the gene $i$ depends by genes in $J$. This information can be used later.

It is now quite easy see that this approach is the same of the previous one, but it take in consideration the dependeces at the same time. For the previous considerations the dependencies between genes at the same time can be found using the primary decomposition apply to the initial timeseries. We have to modify the timeseries in order to use the algorithm 8. For example, if we want to analyze the $i$-th gene, we have to resort all the $s_k$ in order to have the $i$-th component at the end of each time expression. In these condition we are now able to reproduce the algorithm 8 and obtain the dependencies.
Example 11. Using table 3.1 and the gene $g_1$. We have to move the first coordinate at the and and shift the others. The result is

\[
\begin{align*}
    s_1 &= (2, 2, 2) \\
    s_2 &= (0, 2, 1) \\
    s_3 &= (0, 0, 1) \\
    s_4 &= (1, 1, 0) \\
    s_5 &= (1, 1, 0)
\end{align*}
\]

The output is \{x_1\}. This means that $g_1$ at each level can be recovered only by the expression of $g_2$ (note that we have to revert the order of the coordinates and so $x_1$ stands for $g_2$). Combined with the method described before in subsection 5.2.2 we can have more information to use choosing the suitable set of dependence.

This new algorithm can be used simultaneously with the algorithm 8. Thanks to algorithm 8 we obtain the set of the possible minimal subsets. Using this new algorithm we can clump some subsets, apparently different, and obtain a smallest subset. We can figure this in the following example:

Example 12. In this example we will indicate the variables $x_i$ with the index $i$. Let $S$ be a timeseries of 10 genes. Suppose to have the set $W$ of minimal subsets for $f_1$:

\[
W = \{\{2, 3\}, \{2, 4\}, \{2, 7, 8\}, \{3, 7\}, \{5, 10\}\}.
\]

Without more information we can not say if \(f_1 = f_1(x_2, x_3)\) or \(f_1 = f_1(x_5, x_{10})\). Let suppose that at each level we can write \(x_7 = h(x_2, x_4)\) and \(x_8 = g(x_3, x_4)\), e.g. we apply the algorithm 9 for $x_7$ and $x_8$ and we obtain $h$ and $g$ (actually we know that $h$ and $g$ exist but we are not interested to their expression). We can replace 7 with 2 and 4. In the same way we can replace 8 with 3 and 4. With these substitutions we can rewrite $W$ as:

\[
W = \{\{2, 3\}, \{2, 4\}, \{2, 2, 4, 3, 4\}, \{3, 2, 4\}, \{5, 10\}\}
\]

It is clear now that if we write \(f_1 = f_1(x_2, x_3, x_4)\) we are considering the most relevant part of $W$.

With much more variable this algorithm can be used with the others to compare the results. For more details and results see chapter 9.
Chapter 6

Stable border bases

All the previous chapters use a discretization of the time series. We try a different approach using the real value of time expression. We could consider \( \mathbb{R} \) or \( \mathbb{Q} \) and use the same methods illustrated above, with exception to the primary decomposition algorithm (see the construction of the sets \( X_n \)). The problem using Gröbner basis is that we force the function to be polynomial, and this leads problem of overfitting. Anyway we try a new method, based on the stable border bases for vanishing ideals described by Maria Laura Torrente recently in [AF08]. We resume now the basically definitions and some considerations, the implementation of the algorithm can be found in CoCoA [CoC] and more information in the Torrente’s paper.

6.1 Mathematical aspects

Let \( R = \mathbb{K}[x_1, \ldots, x_n] \) a polynomial ring over the field \( \mathbb{K} \), and \( T^n \) be the moniod of power products of \( R \) and \( O \subset T^n \), then:

**Definition 36.** The factor closure of \( O \) is the set \( \overline{O} \) of all power products in \( T^n \) that divide some element of \( O \). If \( O = \overline{O} \) it is called order ideal. Let \( \mathcal{I} \) a 0-dimensional ideal of \( R \). If \( O \) is a order ideal and the residue classes of its elements form a basis of \( R/\mathcal{I} \), then \( O \) is called quotient basis for \( \mathcal{I} \). Moreover, if \( O \) is a order ideal we define its border \( \partial O \) as:

\[
\partial O = (x_1 O \cup \cdots \cup x_n O) \setminus O.
\]

Finally, if \( O \) is a order ideal, then the elements of the minimals set of generators of \( T^n \setminus O \) are called corners of \( O \).

We need some more definition and proposition:
CHAPTER 6. STABLE BORDER BASES

Definition 37. Let $\mathcal{O} = \{t_i\}_{i=1}^\mu$ an order ideal and $\partial\mathcal{O} = \{b_i\}_{i=1}^\nu$ its border. Let $\mathcal{B} = \{g_i\}_{i=1}^\mu$ be a set of polynomials in the form:

$$g_j = b_j - \sum_{i=1}^{\mu} \alpha_{i,j} t_i$$

(6.1)

with $\alpha_{i,j} \in \mathbb{R}$. Let $\mathcal{I} \subset \mathbb{R}$ containing $\mathcal{B}$. We say that $\mathcal{B}$ is a $\mathcal{O}$-border basis of $\mathcal{I}$ if the residue classes of the elements of the order ideal form a real vector space basis of $R \setminus \mathcal{I}$.

In [KR05] we can find the proof that in the condition above, a border basis exists and it is unique. In $\mathbb{R}^n$ we use the euclidean norm $|·|$, moreover, given a $n \times n$ diagonal matrix $E$ we can define a weighted 2-norm as:

$$|v|_E = |Ev|.$$

Definition 38. Let $p \in \mathbb{R}^n$ and $\varepsilon \in \mathbb{R}^n_+$. And empirical point $p^\varepsilon$ is the pair $(p, \varepsilon)$ and we call $p$ the specified value and $\varepsilon$ the tolerance. In this conditions, we define the ellipsoid of perturbation of $p^\varepsilon$:

$$N(p^\varepsilon) = \{q \in \mathbb{R}^n \text{ s.t. } |q - p|_E \leq 1\},$$

where $E$ is the diagonal matrix with the inverse of the components’ the tolerance as elements.

We can generalize this situation by a point to a set:

Definition 39. Let $X^\varepsilon$ a finite set of empirical points, all with the same tolerance $\varepsilon$, we can define an admissible perturbation of $X^\varepsilon$ as the set $\tilde{X} = \{\tilde{p}_1, \ldots, \tilde{p}_n\}$ such that:

$$(\tilde{p}_1, \ldots, \tilde{p}_n) \in \prod_{i=1}^{s} N(p_i^\varepsilon).$$

Finally, two empirical points $p_1^\varepsilon$ and $p_2^\varepsilon$ are said distinct if $N(p_1^\varepsilon) \cap N(p_2^\varepsilon) = \emptyset$

Definition 40. Let $X = \{p_1, \ldots, p_s\} \subset \mathbb{R}^n$. We call the evaluation map the map:

$$\text{eval}_X : \mathbb{R} \rightarrow \mathbb{R}^s$$

$$f \mapsto (f(p_1), \ldots, f(p_s)).$$

We denote $\text{eval}_X(f)$ as $f(X)$. Moreover, let $G = \{g_i\}_{i=1}^k$ we define the evaluation matrix $M_G(X)$ of $G$ associate to $X$ to be the $s \times k$-matrix whose columns are the image of $g_j$ under the evaluation map.
We are now ready to give the more interesting definition:

**Definition 41.** Let \( \mathcal{O} \) be an order ideal, then \( \mathcal{O} \) is said **stable** for \( X^\varepsilon \) if the evaluation matrix \( M_{\mathcal{O}}(\tilde{X}) \) has full rank for each admissible perturbation \( \tilde{X} \) of \( X^\varepsilon \).

**Theorem 10.** Let \( X^\varepsilon \) be a set of \( s \) distinct points. Let \( \mathcal{O} = \{t_i\}_{i=1}^s \) be a quotient basis for \( \mathcal{V}(X) \) stable for \( X^\varepsilon \). Then, for each admissible perturbation \( \tilde{X} \) of \( X^\varepsilon \), the vanishing ideal \( \mathcal{V}(\tilde{X}) \) has a \( \mathcal{O} \)-border basis. Moreover, if \( \partial \mathcal{O} = \{b_i\}_{i=1}^\nu \) is the border of \( \mathcal{O} \), then \( \tilde{B} \) consists of \( \nu \) polynomials like 6.1 where the \( \alpha_{i,j} \) satisfy the linear system:

\[
b_j(\tilde{X}) = \sum_{i=1}^s \alpha_{i,j} t_i(\tilde{X})
\]

The proof of this theorem can be found in [AF08]. From this theorem follows a natural definition:

**Definition 42.** In the hypotheses of the theorem 10 the \( \mathcal{O} \)-border basis \( B \) for \( \mathcal{V}(X) \) is said to be **stable** respect to \( X^\varepsilon \).

We conclude the theory about border basis with an important theorem useful for making an algorithm that finds the border basis:

**Theorem 11.** Let \( X \) a set of distinct points, \( \mathcal{V}(X) \) the vanishing ideal, \( \mathcal{O} \) a quotient basis for \( \mathcal{V}(X) \) then exists a tolerance \( \delta \) such that \( \mathcal{O} \) is stable respect to \( X^\delta \).

The algorithm implemented by the authors of [AF08] can be found in CoCoa System program as the function \( \text{StableBBasis5(Points,Tolerance vector)} \). The power of this algorithm is shown in the follow example.

**Example 13.** Let \( X \subset \mathbb{R}^2 \) be a set of 5 points created perturbing less than 0.01 the ellipse \( X^2 + 0.25Y^2 - 25 = 0 \).

\[
X = \{(-5.07, 0.02), (4.98, 0.0), (3.05, 8.07), (3.01, -8.02), (-3.02, 7.99)\}.
\]

The output of \( \text{StableBBasis}(X,(0.1,0.1)) \) has a minimal degree solution that is:

\[
x^2 - 0.02147711xy + 0.0903012x + 0.2441082y^2 - 0.03758181y - 25.2501,
\]
a very good approximation of the ellipse.

Compare this solution to the vanishing ideal:

\[
\mathcal{V}(X) = \langle y^5 - 8.06y^4 - 64.1611y^3 + 518.4104y^2 - 10.34248y,
\]
\[
x + 0.8992637y^4 - 7.826976y^3 - 57.78237y^2 + 503.6588y - 4.98 \rangle
\]
6.2 Use border bases for reverse engineering

Looking at the timeseries and at the regular trajectories of the gene expressions, we can conjecture that a polynomial solution over $\mathbb{R}$ could be acceptable. The problem is that Buchberger- Möller algorithm is not a stable algorithm. As shown in example 13, if we have real points, generally the vanishing ideal does not recover the regularities of the variety. For this reasons we decide to create a new algorithm that use the potential of the stable border bases. Unfortunately, we learned later about this new instrument, and so we still need to refine this method and we have to investigate more in this area.

The strategy is the same of chapter 5.

Algorithm 10. For each gene $g_i$:

1. Modify the timeseries adding at the end of $s_j$ the $i$-th value of $s_{j+1}$ for all $j < m$ where $m$ is the number of timeseries.

2. Choose a term order in which the last variable is greater then the others, e.g. we work in $\mathbb{Q}[x_1, \ldots, x_n, h]$ there $h$ is the greater variable.

3. Start with tolerance $\delta = (1, \ldots, 1)^1$ and decrease it until $\text{StableBBasis5}$ gives a stable border bases.

4. Look at the $\text{AlmostVanishing}$ polynomials in the output, and find the one, or more polynomials, which have a linear expression in $h$. For the

\footnote{The choice of uniform and unitary vector of tolerance is a methodological choice. Experimentally the dataset goes under this tolerance.}
choice of the term order in the step 2, \( h \) is the first variable appearing in each polynomial in \textit{AlmostVanishing}.

5. Now, if setp 4 successes, we can write \( h = f(x_1, \ldots, x_n) \) and so \( f \) represent the regulator function for \( g_i \).

**Example 14.** We try this algorithm to the dataset 1.3 for the gene \( g_1 \). The timeseries becomes:

<table>
<thead>
<tr>
<th>Time</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( h )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.6104</td>
<td>1.2042</td>
<td>1.0072</td>
<td>1.7073</td>
</tr>
<tr>
<td>2</td>
<td>1.7073</td>
<td>1.3252</td>
<td>1.0185</td>
<td>1.7254</td>
</tr>
<tr>
<td>3</td>
<td>1.7254</td>
<td>1.4118</td>
<td>1.0336</td>
<td>1.7011</td>
</tr>
<tr>
<td>4</td>
<td>1.7011</td>
<td>1.4616</td>
<td>1.0508</td>
<td>1.6601</td>
</tr>
</tbody>
</table>

The next step is to fix a term order \( \prec \) such that \( x_i \prec h \) for all \( i = 1 \ldots 3 \). We choose a weight order with \( u = (1, 1, 1, 100) \) and \textit{Lex} for eventually breaks tie. The step 3 search a suitable tolerance, for these points we find out that \( \delta = (0.006, 0.006, 0.006, 0.006) \) is the maximum of tolerance that can be chosen. With this tolerance the output \textit{AlmostVanishing} of the function \textit{StableBBasis5} is:

\[
0.5694076h + x_3 - 1.994650 \\
-1.336786h + x_1 - 0.5960214x_2 + 1.389493, \\
h^2 - 3.385563h - 0.001695341x_2 + 2.867196, \\
-1.201169x_2h + 2.091681h + x_2^2 - 0.5534222x_2 - 1.885274.
\]

The last two polynomials are useless for our purpose. We focus the attention at the first two and remembering that \( h \) indicates the function \( f_1 \), we can rewrite them as:

\[
f_1 = -0.5694076x_3 + 1.135768, \\
f_1 = 1.336786x_1 - 0.796753x_2 + 1.857455.
\]

We recognize the dependence of the gene \( g_1 \). As we see in 5.1 and 4.6 we can suppose that \( g_1 \) depends only by \( g_3 \).

This algorithm can be used together with the primary decomposition and the Gröbner fan method in order to compare the results.
Chapter 7
Linear codes

Let now see some aspects of linear codes. An other algebraic approach for reconstruction of gene regulatory networks is based on the Reed-Muller codes. This method is shown by Dingel and Milenkovic in [DM09] and [DM08]. We propose now the background to understand it and a brief for an other possible method using code theory.

7.1 Mathematical aspects

Definition 43. Let $\mathbb{F}_q$ be a finite field with $q = p^m$ elements. Let $k, n \in \mathbb{N}$ such that $k \leq n$. A linear code $C$ is a $k$-dimensional vector subspace of $(\mathbb{F}_q)^n$. We call $k$ the dimension and $n$ the length. An element of $C$ is called word.

Obviously a linear code $C$ like definition 43 has $q^k$ elements.

We can introduce a distance over $(\mathbb{F}_q)^n$ calling Hamming distance defining by:

Definition 44. Let $x, y \in (\mathbb{F}_q)^n$, the Hamming distance between $x$ and $y$:

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

i.e. the number of component where the two word differ.

Definition 45. Let $x \in (\mathbb{F}_q)^n$. We define the Hamming weight of $x$, $w(x) = d(x, 0)$.

Definition 46. Let $C$ a linear code. We define the distance of $C$ as the minimum Hamming distance between any two words.

Since $C$ is a lineal vector space, the distance of $C$ is the minimum of Hamming weight of its non-0 words.
Definition 47. A \([n, k, d]\) code is a linear code of dimension \(k\), length \(n\) and distance \(d\).

A code \(C\) can be used to send some message in a safe way, where safe means that we can recover the message from some transmission error. This operation is divided into parts

- An invertible algorithm called coding procedure whose input is a \(k\)-symbol vector and the output is a \(n\)-symbol vector i.e. a word of \(C\).
- We split the message into information data and transmit it into a noisy channel one at time.
- We invert the coding procedure and we try to rebuild the original message with the decoding procedure. In this last step we can obtain two type of output:
  - the sent message if there are a few or none errors.
  - a message of error if there are unrecoverable errors.

One of the most interesting part is the detection procedure. A procedure that detect errors without trying to correct them. If there is a detection (decoding) procedure that detect \(s\) errors or less, than we say that the error detection capability of the code \(C\) is at least \(s\). If no other procedure can do better, then the error detection capability is exactly \(s\). In the same way if there is a procedure that correct \(t\) errors, then we say that the error correction capability of the code \(C\) is at least \(t\). There is an important theorem for the \([n, k, d]\) codes:

Theorem 12. An \([n, k, d]\) code over \((\mathbb{F}_q)^n\) has detection capability \(d - 1\) and correction capability \(t = \lfloor \frac{d-1}{2} \rfloor\).

When we send a message as a vector, the channel in which it is transmitted can modify it. In other words what we send is not what is received. In this situation can happen that the channel modify a bit, a entire of the vector, or some of them. In this case we call it error. The correction capability is \(t\) as before. When the decoder is more sophisticated it can return a unknown value if it is not sure of it. In this case we do not have a wrong value but a missing one. In this case we call it erasure. It is easy to see that in this situation the correction capability is \(d - 1\).

Since \(C\) is a vector subspace, we can represent it by a matrix formed by a minimum set of its generators. This matrix is called the generator matrix of \(C\) and is traditionally denoted by \(\mathcal{G}\). The matrix \(\mathcal{G}\) is a \(k \times n\) matrix with coefficient in \(\mathbb{F}_q\). Give a generator matrix \(\mathcal{G}\), the encoding of a message \(m \in (\mathbb{F}_q)^k\) is \(c = m\mathcal{G}\), as obviously \(c\) is a word of \(C\).
Theorem 13. Give a code $C$ of length $n$ and dimension $k$ then we have this bound:

$$d_{\text{min}} \leq n - k + 1$$  \hspace{1cm} (7.1)

Proof. This is the important **Singleton bound** and a proof can be found in [MS97].

If in 7.1 hold the equality, the code is called **MDS** (Maximum distance separable).

An other important matrix associate to a linear code $C$, is the so-called **parity-check** matrix $H$. This matrix $H \in \mathbb{M}(n-k, n, \mathbb{F}_q)$ is build so that its coefficients are the coefficients of a linear system of equation which solution are the codewords, in other term, $c \in C \iff cH^T = 0$.

7.1.1 Cyclic, Reed-Solomon and Reed-Muller codes

Definition 48. A linear code $C$ is calling cyclic if:

whenever $(c_0, c_1, \ldots, c_{n-1}) \in C$ then $(c_1, c_2, \ldots, c_{n-1}, c_0) \in C$.

There is a algebraic way to looking at this important type of codes. Let $C$ be a cyclic code over $\mathbb{F}_q$ with length $n$ and $n, q$ coprime. We set $R_n = \mathbb{F}_q[x]/(x^n - 1)$. Then we can think that in $R_n$ there are all polynomials of degree at most $n - 1$ and so we can see $C$ as the vector of the coefficient of these polynomial, more specifically we can identify $C$ with:

$$I_C = \{ c(x) = \sum_{i=0}^{n-1} c_i x^i \in R_n : c = (c_0, \ldots, c_{n-1}) \in C \}$$

Moreover, we have that $I_C$ is an ideal of $R_n$. Let $C$ a $[n, k, d]$ linear code, then a unique monic polynomial $(g(x))$ of minimal degree $(n - k)$ that generate $I_C$ and this polynomial divides $x^n - 1$. The proof of these facts can be found in [Wal00].

Definition 49. Let $\{\alpha_1, \ldots, \alpha_n\} \subset \mathbb{F}_q$ $n$ distinct points (evaluation points) of the finite field with $q$ elements and choose $n, k$ such that $k \leq n \leq q$. We define an encoding function for **Reed-Solomon Code** as:

$$RS : \mathbb{F}_q^k \rightarrow \mathbb{F}_q^n$$

$$m = (m_0, \ldots, m_{k-1}) \mapsto f_m(x)$$

where

$$f_m(x) = \sum_{i=0}^{k-1} m_i x^i$$

The encoding of $m$ is the evaluation of $f_m(x)$ at all the $\alpha_i$. 
It is easy to see that a RS code is a \([n, k, n - k + 1]\) MDS linear code. We want to extend this construction using multivariate polynomial.

**Definition 50.** Let \(F_q[x_1, \ldots, x_n]\) the ring of multivariate polynomials over \(F_q\). Choose \(t\), and we define an Reed-Muller Code \(RM_{q,t}\) giving the matrix \(G\) as follows:

- Let \(W = \{1, x_1, \ldots, x_n, x_1x_2, \ldots, x_{n-1}x_n, \ldots, x_1^t, \ldots, x_n^t, \ldots\}\) to be the set of all monomials of degree \(\leq t\).
- We sort the element of \(F_q = \{\alpha_1, \ldots, \alpha_q\}\). For each \(\alpha_i\) we define the \(i\)-th column as the evaluation of \(p\) on \(\alpha_i\) for each \(p \in W\).

### 7.2 Use codes for reverse engineering

Generally, a timeseries has a lot of variables (genes) and few timesteps. This means that the timeseries, looked as a set of points of \(K^n\), is a little set. It is easy to see that few points means a lot of possible polynomials (or functions, it depends on \(K\)) that do what we want. The idea to apply coding theory arise from the capacity of detection and correction of errors and erasures of the code. We can think that missing data are erasure and so we can, under some conditions, recover them and obtain a unique result of the problem (or to reduce the number of solutions).

We will see in the next example that Reed-Muller codes are the natural choice for this type of problem.

**Example 15.** We use the time series in table 3.1. We have to choose the degree of the polynomials \(t\). As we see in the example 8 the maximum degree of the polynomials is 2, and so we choose \(t = 2\). Moreover, if we work, as in this example, with characteristic 3, has no sense use a higher degree. We can create a generating matrix using the definition of the Reed-Muller code.

The timeseries lives in \((F_3)^3\) and so the polynomial \(f_i\) lives in \(F_3[x_1, x_2, x_3]\). We consider a basis of the linear space of the polynomials of degree \(\leq 2\) e.g. the follow:

\[
B = \{1, x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1^2, x_2^2, x_3^2\}.
\]

We evaluate these polynomials over all points of \(F_3^3\) order in this way:

\[
\begin{align*}
\alpha_1 &= (1, 0, 0) & \alpha_2 &= (2, 0, 0) & \alpha_3 &= (0, 1, 0) & \alpha_4 &= (0, 2, 0) \\
\alpha_5 &= (0, 0, 1) & \alpha_6 &= (0, 0, 2) & \alpha_7 &= (1, 1, 0) & \alpha_8 &= (1, 2, 0) \\
\alpha_9 &= (2, 1, 0) & \alpha_{10} &= (2, 2, 0) & \alpha_{11} &= (0, 1, 1) & \alpha_{12} &= (0, 1, 2) \\
\alpha_{13} &= (0, 2, 1) & \alpha_{14} &= (0, 2, 2) & \alpha_{15} &= (1, 0, 1) & \alpha_{16} &= (1, 0, 2) \\
\alpha_{17} &= (2, 0, 1) & \alpha_{18} &= (2, 0, 2) & \alpha_{19} &= (1, 1, 1) & \alpha_{20} &= (1, 1, 2) \\
\alpha_{21} &= (1, 2, 1) & \alpha_{22} &= (1, 2, 2) & \alpha_{23} &= (2, 1, 1) & \alpha_{24} &= (2, 1, 2) \\
\alpha_{25} &= (2, 2, 1) & \alpha_{26} &= (2, 2, 2) & \alpha_{27} &= (0, 0, 0)
\end{align*}
\]
and we build $G$:

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

This is a $[27, 10, 9]$ linear code.

For example we try to recover $f_1(x_1, x_2, x_3)$. We know that $f_1(2, 2, 2) = 1, f_1(1, 0, 2) = 1, f_1(1, 0, 0) = 0$ and $f_1(0, 1, 1) = 0$. We have supposed that the polynomials have degree 2. For this reason $f_1$ is a $\mathbb{F}_3$-linear combination of the monomials of $B$. The linear dependence can be seen as a combination of row of the matrix $G$. We know four value of the evaluation of $f_1$ over $(\mathbb{F}_3)^3$. The distance $d = 9$ say us that we can rebuild $f_1$ if 8 or less evaluation of $f_1$ miss (erasures).

Generally, we have few timesteps and a lot of variables. The recovery property of codes allow to reduce the loss of information. The algorithm that uses Reed-Muller codes can be found in [DM09]. For reason of space and time we do not focus the attention on this approach. The algorithm in [DM09] has as output the maximum degree of monomials that can be found using a decoding radius. The recovery of the erasures can be used the method described in [Nan07] a modification of the list decoding shown by Pellikaan and Wu in [PW04].

### 7.2.1 Possible improvements

In this last subsection we want to explain how to improve the algorithm cited above.

**Definition 51.** Let $C$ a $[n, k, d]$ linear code. We define the weight distribution of $C$ as a vector $(A_0, \ldots, A_n)$ such that $A_i = |\{c \in C \text{ s.t. } w(c) = i\}|$. It is obvious that $A_0 = 1$ and $A_i = 0$ for all $1 \leq i \leq d - 1$.

The weight distribution says how many words of length $k$ there are in the code $C$. We want to use the weight distribution in order to reduce the number of possible $f_i$. 
Example 16. Referring to the code in example 15 we can summarize its weight distribution:

\[ A_0 = 1, \ A_1 = A_2 = A_3 = A_4 = A_5 = A_6 = A_7 = A_8 = 0, \]
\[ A_9 = 78, \ A_{10} = A_{11} = 0, \ A_{12} = 1404, \ A_{13} = A_{14} = 0, \]
\[ A_{15} = 14040, \ A_{16} = A_{17} = 0, \ A_{18} = 27300, \ A_{19} = A_{20} = 0, \]
\[ A_{21} = 15444, \ A_{22} = A_{23} = 0, \ A_{24} = 702, \ A_{25} = A_{26} = 0, \ A_{27} = 80. \]

We can figure out the information about \( f_1 \) using its evaluation in the four point:

\[ f_1^T = (0, *, *, *, *, *, *, *, *, *, 0, *, *, *, *, 1, *, *, *, *, *, *, *, *, *, 1, *) \]

where * figure the erasures. In this example we can assume that \( f_1 \) as a word of the code, has not weight exactly 9 (the minimal word are only 78). With this supposition, \( f_1 \) have at least distance equal to 12 because \( A_{10} = A_{11} = 0 \). Moreover we can suppose that \( w(f_1) = 15, 18 \) or 21 according to the weight distribution. This supposition reduced the possibilities of \( f_1 \) and maybe, with future studies, increase the correction capability of erasures. An other possible approach is to minimize, when we correct erasures, the number of variables. We do not focus this work in this direction, and more future studies can be useful to determinate the power of this approach.
Chapter 8

Finite field method

The aim of all this work is to find the polynomials $f_1, \ldots, f_n$ which interpolate the time series and, since there are more than one expression for these polynomials, we have to find a way to choose the representation which can rebuilt the gene network. We saw a method that require Gröbner basis and the reduction of particular solution, and an other one that uses the primary decomposition. We can choose an other way: since each polynomials $f_1, \ldots, f_n$ takes input in $(\mathbb{F}_q)^n$ and returns a value in $\mathbb{F}_q$, we can consider a polynomial $F$ in one variable, from $\mathbb{F}_q^n$ into $\mathbb{F}_q^n$. For simplicity we can choose $q = p$ a prime. Let

$$\psi : (\mathbb{F}_p)^n \to \mathbb{F}_p^n$$

$$(a_1, \ldots, a_n) \mapsto a_1\alpha^{n-1} + a_2\alpha^{n-2} + \ldots + a_n$$

Where $\alpha$ is a primitive element of $\mathbb{F}_p^n$, or a root of the primitive irreducible polynomial defining $\mathbb{F}_p^n$ as in subsection A.3. It is easy to see that $\psi$ is a isomorphism for all seen in subsection A.3. Moreover in $\mathbb{F}_p^n$, that is a field, there is the multiplication. Then if we have $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in (\mathbb{F}_p)^n$ we can consider the product of their image

$$(a_1\alpha^{n-1} + a_2\alpha^{n-2} + \ldots + a_n)(b_1\alpha^{n-1} + b_2\alpha^{n-2} + \ldots + b_n) = c_1\alpha^{n-1} + c_2\alpha^{n-2} + \ldots + c_n.$$

The coefficients $c_i$ are uniquely determinate and so we can pullback the multiplication

$$(a_1, \ldots, a_n) \cdot (b_1, \ldots, b_n) := \psi^{-1}(c_1, \ldots, c_n).$$

8.1 A new algorithm

We want to use this morphism between $(\mathbb{F}_q)^n$ and $\mathbb{F}_q^n$ in order to create a new algorithm. The Gröbner fan approach shows the problem of the choice
of the term order when we reduce the particular solution. Using the primary
decomposition and the double primary decomposition, we can reduce the
number of possible models. Unfortunately the number possible polynomials
can be very huge. In order to avoid this problem, we try a different approach.
We saw before that in $K[x]$, a polynomial ring in just one variable, we have no
problem when we apply the Euclidean algorithm between two polynomials.
We want to use this fact and the isomorphism shown above in order to create
a new method.

**Algorithm 11.** With the previous notation:

- choose a representation of $\mathbb{F}_{p^n}$, i.e. a polynomial of degree $n$ irreducible,
  monic and primitive (we have a lot of choice as formula in definition 70).

- Write the point of the time series as a element in $\mathbb{F}_{p^n}$ using the map $\psi$
defined in above.

- Compute, using Lagrange interpolation, the polynomial in one variable
  $F(X) \in \mathbb{F}_{p^n}[X]$.

- Compute the vanishing ideal $\mathcal{J}$ and reduce $F(X)$ by it. Note that now
  we are in one variable and so the reduction is unique.

- Finally, obtain the polynomial $f_1, \ldots, f_n$ using the pullback of the mul-
tiplication.

We note that also in this case the final result is not unique: it depends
from the choice of the polynomial representing the field $\mathbb{F}_{p^n}$, and the order
of variables. Indeed we can permute the coordinates and we obtain probably
a different solution.

**Example 17.** Referring to time series 3.1 we choose $f(X) = X^3 + 2X + 1$
an irreducible,monic and primitive polynomial of degree 3 over $\mathbb{F}_3$ between
the $\frac{2(3^3-1)}{3} = \frac{12}{3} = 4$ possible. We choose the natural order for the variables.
Let $\alpha$ a primitive root of $f(X)$. We rewrite the time series:

\[
\begin{align*}
p_1 &= (2, 2, 2) = 2\alpha^2 + 2\alpha + 2 \\
p_2 &= (1, 0, 2) = \alpha^2 + 2 \\
p_3 &= (1, 0, 0) = \alpha^2 \\
p_4 &= (0, 1, 1) = \alpha + 1 \\
p_5 &= (0, 1, 1) = \alpha + 1
\end{align*}
\]
8.1. A NEW ALGORITHM

We use the Lagrange interpolation with a little modification due to a double point, and we obtain:

\[ F(X) = 2X^4 + (2\alpha^2 + \alpha)X^3 + 2\alpha^2X^2 + (2\alpha + 1)X + \alpha \]

The vanishing idea, in one variable is just the product of \((X - p_i)\) where \(p_i\) are distinct, and so we have

\[ \mathcal{J} = \langle X^4 + (2\alpha^2 + 1)X^3 + \alpha X^2 + (2\alpha^2 + \alpha + 1)X + 2\alpha + 1 \rangle \]

And so reducing we obtain

\[ F(X) = (\alpha^2 + \alpha + 1)X^3 + (2\alpha^2 + \alpha)X^2 + (2\alpha^2 + 2)X + 1. \]

Now we have to extrapolate the three polynomials. We remember that the variable \(X\) is an element of \(F_{27}\). For this reason is in the form \(a_1\alpha^2 + a_2\alpha + a_3\) with \(a_i \in F_3\) thanks to the isomorphism 8.1. For this reason we can rewrite \(X = (x_1, x_2, x_3)\) where \(x_i\) are the variable for the \(a_i\). Now we have to compute

\[ X^3 = X \cdot X \cdot X = (x_1, x_2, x_3) \cdot (x_1, x_2, x_3) \cdot (x_1, x_2, x_3) \]

for this we use the definition of the product, as the preimage of the product in the field \(F_{27}\):

\[ (x_1\alpha^2 + x_2\alpha + x_3)^3 = x_1^3\alpha^6 + x_2^3\alpha^3 + x_3^3 = (\alpha^2 + \alpha + 1)x_1 + (\alpha + 2)x_2 + x_3 \]

we use that we are in characteristic 3. In the same way we can write that

\[ X^2 = (x_1\alpha^2 + x_2\alpha + x_3)(x_1\alpha^2 + x_2\alpha + x_3) = (\alpha^2 + 2\alpha)x_1^2 + \alpha^2 x_2^2 + x_3^2 + (2\alpha + 1)x_1x_2 + 2\alpha x_1x_3 + 2\alpha x_2x_3 \]

Now, for convenience, we write all the elements of \(F_{27}\) as power of \(\alpha\) (using the fact that \(F_{27}^*\) is cyclic and generated by \(\alpha\)).

\[
\begin{align*}
F(X) &= \alpha^6X^3 + \alpha^{17}X^2 + \alpha^8X + 1 \\
F(x_1, x_2, x_3) &= \alpha^6(x^6x_1 + \alpha^3x_2 + x_3) + \alpha^{17}(\alpha^4x_1^2 + \alpha^2x_2^2 + x_3^2 + \alpha^6x_1x_2 + \\
&\quad + \alpha^{15}x_1x_3 + \alpha^{14}x_2x_3) + \alpha^8(\alpha^2x_1 + \alpha x_2 + x_3) + 1 \\
&= \alpha^5x_1 + \alpha^{22}x_2 + \alpha x_3 + \alpha^{21}x_1^2 + \alpha^{19}x_2^2 + \alpha^{17}x_3^2 + \alpha^7x_1x_2 + \\
&\quad + \alpha^6x_1x_3 + \alpha^5x_2x_3 + 1 \\
&\text{we return now to the standard notation} \\
&= (2\alpha^2 + \alpha + 2)x_1 + (2\alpha + 2)x_2 + \alpha x_3 + (\alpha^2 + 1)x_1^2 + (2\alpha^2 +}
\]
\begin{align*}
+2\alpha + 2)x_2^2 + (2\alpha^2 + \alpha)x_3^2 + (\alpha^2 + 2\alpha + 2)x_1x_2 + (\alpha^2 + \alpha + 1)x_1x_3 + (2\alpha^2 + \alpha + 2)x_2x_3 + 1 \\
= \alpha^2(2x_1 + x_1^2 + 2x_2^2 + 2x_3 + x_1x_2 + x_1x_3 + 2x_2x_3) \\
\quad + \alpha(x_1 + 2x_2 + x_3 + 2x_2^2 + x_3^3 + 2x_1x_2 + x_1x_3 + x_2x_3) \\
\quad + 1(2x_1 + 2x_2 + x_1^2 + 2x_2^2 + 2x_1x_2 + x_1x_3 + 2x_2x_3 + 1) \\
= \alpha^2f_1(x_1, x_2, x_3) + \alpha f_2(x_1, x_2, x_3) + f_3(x_1, x_2, x_3).
\end{align*}

In the last form, we clearly see how to rewrite \( F(X) \) in terms of \( x_1, x_2, x_3 \), and we obtain the three polynomial:

\begin{align*}
  f_1(x_1, x_2, x_3) &= x_1^2 + 2x_2^2 + 2x_3 + x_1x_2 + x_1x_3 + 2x_2x_3 + 2x_1 \\
  f_2(x_1, x_2, x_3) &= 2x_2^2 + x_3 + 2x_1x_2 + x_1x_3 + x_2x_3 + x_1 + 2x_2 + x_3 \\
  f_3(x_1, x_2, x_3) &= x_1^2 + 2x_2^2 + 2x_1x_2 + x_1x_3 + 2x_2x_3 + 2x_1 + 2x_2 + 1
\end{align*}

This is a new approach for the problem show until now. We do not find any implementation of a such complex literal calculus, and so in the final chapter this method is not used.
Chapter 9

Test

In this chapter we will use some methods described in the previous chapters in order to reconstruct a numerical GRN. We generated a synthetic time-series using the simulator *netsim* developed by Di Camillo at all [dCTC09]. This simulator is implemented as an R-script; for our simulation we use the following parameters:

- \( N=6 \), the number of genes,
- \( connectivity=c(\"random\") \) a parameter for the topology of the network,
- \( gamma=2.2, \text{INdegree}=c(\"out\"), kappa=2, act.fun=c(\"linear\")\),
- \( max.reg=2 \), maximum regulators for gene,
- \( num.subnet=c(1,1,1) \), number of modules for each type,
- \( times=\text{seq}(0,1,0.07) \), the initial and final time and the temporal step (and so the number of timeseries).

For more details regarding parameters see [dCTC09]. The next step is to discretize these data. We do not use the discretization algorithm proposed by Dimitrova described in subsection 3.1.1 but the linear discretization in 3.1.2. We list the topology of the network (figure 9.1) the simulated timeseries (table 9.1) and three levels of discretization (table 9.2). In figure 9.2 we list the gene expression’s curves, and their approximation with \( p = 7 \).
Figure 9.1: The network given by the synthesizer seen as a directed graph

<table>
<thead>
<tr>
<th></th>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$g_3$</th>
<th>$g_4$</th>
<th>$g_5$</th>
<th>$g_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>3.362958</td>
<td>1.684060</td>
<td>4.235801</td>
<td>8.831972</td>
<td>7.217909</td>
<td>2.781088</td>
</tr>
<tr>
<td>$t_2$</td>
<td>3.452234</td>
<td>1.836343</td>
<td>4.680308</td>
<td>8.391801</td>
<td>6.856191</td>
<td>3.188346</td>
</tr>
<tr>
<td>$t_3$</td>
<td>3.568927</td>
<td>1.937350</td>
<td>5.090567</td>
<td>8.026728</td>
<td>6.553452</td>
<td>3.555783</td>
</tr>
<tr>
<td>$t_4$</td>
<td>3.705913</td>
<td>1.996536</td>
<td>5.469186</td>
<td>7.722113</td>
<td>6.300114</td>
<td>3.889814</td>
</tr>
<tr>
<td>$t_5$</td>
<td>3.858461</td>
<td>2.026565</td>
<td>5.817361</td>
<td>7.468182</td>
<td>6.089505</td>
<td>4.194832</td>
</tr>
<tr>
<td>$t_6$</td>
<td>4.020678</td>
<td>2.017980</td>
<td>6.135872</td>
<td>7.252719</td>
<td>5.913137</td>
<td>4.476382</td>
</tr>
<tr>
<td>$t_7$</td>
<td>4.188466</td>
<td>1.995046</td>
<td>6.428940</td>
<td>7.067086</td>
<td>5.764722</td>
<td>4.738457</td>
</tr>
<tr>
<td>$t_8$</td>
<td>4.359495</td>
<td>1.955789</td>
<td>6.702253</td>
<td>6.906213</td>
<td>5.640187</td>
<td>4.983506</td>
</tr>
<tr>
<td>$t_9$</td>
<td>4.531783</td>
<td>1.903732</td>
<td>6.956427</td>
<td>6.765748</td>
<td>5.535889</td>
<td>5.213751</td>
</tr>
<tr>
<td>$t_{10}$</td>
<td>4.703701</td>
<td>1.841928</td>
<td>7.191157</td>
<td>6.642040</td>
<td>5.448626</td>
<td>5.431116</td>
</tr>
<tr>
<td>$t_{11}$</td>
<td>4.874052</td>
<td>1.772843</td>
<td>7.407759</td>
<td>6.532108</td>
<td>5.375701</td>
<td>5.637074</td>
</tr>
<tr>
<td>$t_{12}$</td>
<td>5.041733</td>
<td>1.698825</td>
<td>7.607568</td>
<td>6.433165</td>
<td>5.314568</td>
<td>5.830108</td>
</tr>
<tr>
<td>$t_{13}$</td>
<td>5.206105</td>
<td>1.621658</td>
<td>7.791932</td>
<td>6.343403</td>
<td>5.263435</td>
<td>6.019945</td>
</tr>
<tr>
<td>$t_{14}$</td>
<td>5.365666</td>
<td>1.543018</td>
<td>7.962118</td>
<td>6.261079</td>
<td>5.220552</td>
<td>6.198812</td>
</tr>
<tr>
<td>$t_{15}$</td>
<td>5.522889</td>
<td>1.463928</td>
<td>8.119177</td>
<td>6.185171</td>
<td>5.184685</td>
<td>6.370212</td>
</tr>
</tbody>
</table>

Table 9.1: Synthetical gene expression timeseries
Table 9.2: (a) Discretization with $p = 3$. (b) Discretization with $p = 5$. (c) Discretization with $p = 7$.

Figure 9.2: Left pannel: real data. Right pannel: discretized data on $F_7$. 
9.1 Gröbner fan and primary decomposition results

For each discretization we first calculate all possible function $f_i$ using the Gröbner fan method described in section 4.3. We apply the primary decomposition algorithm shown in chapter 5. We use the same discretization used in the Gröbner fan method in the previous section. We have to modify all tables in 9.2 in order to add to every timestep $s_i$ the coordinate $s_{i+1,j}$ where $j$ is the gene that we want to study. In the following subsection we will give the primary decomposition relative to the gene $g_1$. Moreover, we use the score indicators introduced in chapter 5, more specifically $T_1$ in example 5.5 and $S_3$ in example 5.4. We use Singular and Sage together because the function groebner_fan is implemented in Sage and the function vanishId in Singular. The first is written in Python and the second one in a pseudo C code. The merge of the two part of the code can be done, with some difficulties, using some libraries of Sage and Python.

$p=3$

We find 36 different classes in the Gröbner fan relative to the vanishing ideal of the points 9.2. We calculate all possible reduction for $f_2$ and we resume them in table 9.3.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$f_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$x_2x_6 - x_3x_6 - x_6^2 + x_2 + x_5 + x_6 - 1$</td>
</tr>
<tr>
<td>2</td>
<td>$x_2x_6 - x_4x_6 - x_6^2 + x_2 + x_3 + x_6 - 1$</td>
</tr>
<tr>
<td>3</td>
<td>$-x_2x_5 + x_5^2 + x_5x_6$</td>
</tr>
<tr>
<td>5</td>
<td>$-x_2x_5 + x_3x_5 + x_5^2$</td>
</tr>
<tr>
<td>4</td>
<td>$-x_2x_4 + x_4^2 + x_4x_6$</td>
</tr>
<tr>
<td>6</td>
<td>$x_2x_3 - x_3^2 - x_3x_5 + x_2 + x_3 + x_5 - 1$</td>
</tr>
<tr>
<td>2</td>
<td>$x_2^2 + x_2x_4 + x_4x_6 - x_1 - x_2 - x_4 - 1$</td>
</tr>
<tr>
<td>1</td>
<td>$-x_2^2 + x_6^2 + x_1 + x_2 + x_5$</td>
</tr>
<tr>
<td>1</td>
<td>$-x_2^2 + x_6^2 + x_1 + x_2 + x_4$</td>
</tr>
<tr>
<td>2</td>
<td>$-x_2^2 + x_3^2 + x_1 + x_2 + x_5$</td>
</tr>
<tr>
<td>6</td>
<td>$x_1x_2 - x_1x_6 + x_2 + x_3 + x_6 - 1$</td>
</tr>
<tr>
<td>2</td>
<td>$x_1x_2 - x_1x_3 + x_2 + x_3 + x_4 - 1$</td>
</tr>
</tbody>
</table>

Table 9.3: All possible reduction of $f_2$ with $p = 3$

As we say before, we are not interested to the polynomials but only to the dependencies. For this reason we clump together the polynomials with the
same variables and we summarize in table 9.4 all the possible combinations. It is easy to see that the subset that appears more frequently is \( \{x_2, x_3, x_5\} \).

<table>
<thead>
<tr>
<th>n</th>
<th>subset of ( {x_1, \ldots, x_6} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>( {x_2, x_5, x_6} )</td>
</tr>
<tr>
<td>6</td>
<td>( {x_2, x_4, x_6} )</td>
</tr>
<tr>
<td>11</td>
<td>( {x_2, x_3, x_5} )</td>
</tr>
<tr>
<td>7</td>
<td>( {x_1, x_2, x_5, x_6} )</td>
</tr>
<tr>
<td>3</td>
<td>( {x_1, x_2, x_4, x_6} )</td>
</tr>
<tr>
<td>2</td>
<td>( {x_1, x_2, x_3, x_5} )</td>
</tr>
<tr>
<td>2</td>
<td>( {x_1, x_2, x_3, x_4} )</td>
</tr>
</tbody>
</table>

Table 9.4: Possible dependencies of \( f_2 \) with \( p = 3 \)

And so if we use table 9.4 for the reconstruction of \( f_2 \) we are adding only the link \( 5 \rightarrow 2 \).

The two primary decomposition for \( g_2 \) and the scores are resumed in table 9.5.

<table>
<thead>
<tr>
<th>decomposition</th>
<th>( T_1, S_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {x_2, x_6} )</td>
<td>2</td>
</tr>
<tr>
<td>( {x_2, x_3} )</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 9.5: Primary decomposition for \( g_2 \) with \( p = 3 \)

Comparing tables 9.4 and 9.5 we can see that the added link \( 5 \rightarrow 2 \) can be removed. Indeed using the primary decomposition we note that the variable \( x_5 \) is not useful to determinate \( f_2 \). It is important to note that we can not use only table 9.5 for recovering \( f_2 \), indeed we can not choose \( \{x_2, x_6\} \) in place of \( \{x_2, x_3\} \). We want to remark that comparing the two table we can recover \( f_2 \). Sometimes these two tables are not sufficient. In this case we can use also the double primary decomposition or some biological informations. We will see an example of this in the last section of this chapter.
\[ p = 5 \]

We will give the corresponding results for \( p = 5 \). The number of classes in this case grows to 1784. Also in this case we will consider only the possible reduction of \( f_2 \) summarized in Table 9.6.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( f_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>( 2xx2x5 + 2xx3x6 - x_1 - x_3 + 2x_5 + x_6 + 2 )</td>
</tr>
<tr>
<td>35</td>
<td>( 2xx2x5 - 2xx3x5 - 2x_5^2 - 2x_5x_6 - x_1 + x_5 - x_6 - 2 )</td>
</tr>
<tr>
<td>23</td>
<td>( 2xx2x4 - 2xx3x4 + x_4^2 + 2xx4x5 + 2xx4x6 - x_1 + x_5 - x_6 - 2 )</td>
</tr>
<tr>
<td>20</td>
<td>( 2x_2^2 - 2xx2x4 + 2xx1x6 - 2x_2 - 2x_3 - 2x_4 - 2x_6 - 1 )</td>
</tr>
<tr>
<td>85</td>
<td>( x_1x_2 - 2xx2x4 - 2x_2^2 + 2x_1 + x_3 + x_5 - x_6 )</td>
</tr>
<tr>
<td>51</td>
<td>( -2x_3^3 + x_2^2 - 2xx4x5 + 2x_4^2 + x_1 - x_2 - x_3 + 2x_4 + x_5 - x_6 - 1 )</td>
</tr>
<tr>
<td>62</td>
<td>( -2x_4^3 + x_2^2 + x_1 - x_2 - x_3 - x_4 + x_5 - x_6 - 1 )</td>
</tr>
<tr>
<td>31</td>
<td>( 2x_2^3 + 2x_2x^2 + x_2x_3 - 2xx2x4 - 2x_2^2 + 2x_2 + x_3 - x_4 - x_5 - x_6 )</td>
</tr>
<tr>
<td>21</td>
<td>( -2x_1^3 + 2xx_1x_5 - 2x_1^2 + 2xx_2x_5 + x_5^2 + 2x_1 + 2x_5 - x_6 + 1 )</td>
</tr>
<tr>
<td>23</td>
<td>( 2x_3^2 - 2x_4^3 - 2xx_4x_5 + 2xx_4x_6 - x_1 + 2x_5 - x_6 - 2 )</td>
</tr>
<tr>
<td>20</td>
<td>( -x_2^4 + 2xx_2x_3 + 2x_2^2 - 2xx_4 - 2xx_2x_5 - 2x_5^2 - x_2 - x_4 + x_5 - x_6 - 1 )</td>
</tr>
</tbody>
</table>

Table 9.6: Some possible reduction of \( f_2 \) with \( p = 5 \)

The others polynomial appears with frequency less than 19 and for reason of space we do not report them. Table 9.7 shows the dependencies:

---

Note: Before we cannot look at \( f_1 \) because there is not a particular solution of it. It arise from the fact that the timeseries with \( p = 3 \) does not lead a possible function \( f_1 \). In order to compare the results, we will use \( f_2 \) in each experiment.
9.1. GRÖBNER FAN AND PRIMARY DECOMPOSITION RESULTS

<table>
<thead>
<tr>
<th>n</th>
<th>subset of {x_1, \ldots, x_6}</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>{x_2, x_4, x_6}</td>
</tr>
<tr>
<td>141</td>
<td>{x_2, x_4, x_5, x_6}</td>
</tr>
<tr>
<td>6</td>
<td>{x_2, x_3, x_4, x_6}</td>
</tr>
<tr>
<td>146</td>
<td>{x_2, x_3, x_4, x_5, x_6}</td>
</tr>
<tr>
<td>29</td>
<td>{x_1, x_2, x_6}</td>
</tr>
<tr>
<td>187</td>
<td>{x_1, x_2, x_5, x_6}</td>
</tr>
<tr>
<td>25</td>
<td>{x_1, x_2, x_4, x_6}</td>
</tr>
<tr>
<td>266</td>
<td>{x_1, x_2, x_4, x_5, x_6}</td>
</tr>
<tr>
<td>83</td>
<td>{x_1, x_2, x_3, x_6}</td>
</tr>
<tr>
<td>320</td>
<td>{x_1, x_2, x_3, x_5, x_6}</td>
</tr>
<tr>
<td>37</td>
<td>{x_1, x_2, x_3, x_4, x_6}</td>
</tr>
<tr>
<td>523</td>
<td>{x_1, x_2, x_3, x_4, x_5, x_6}</td>
</tr>
</tbody>
</table>

Table 9.7: Possible dependencies of \(f_2\) with \(p = 5\)

The two primary decomposition for \(g_2\) and the scores are resumed in table 9.8.

<table>
<thead>
<tr>
<th>decomposition</th>
<th>(T_1, S_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{x_2, x_4, x_6}</td>
<td>4</td>
</tr>
<tr>
<td>{x_1, x_2, x_6}</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 9.8: Primary decomposition for \(g_2\) with \(p = 5\)

\(p = 7\)

With \(p = 7\) the number of elements of the Gröbner fan is 3375. Some possible reduction of \(f_2\) are summarize below in table 9.9, also in this case we will show only the most frequent polynomials:

In table 9.10 we can see all the possible dependencies.

In table 9.11 we summarize the four possible minimal subset for \(f_2\) using the primary decomposition.
Table 9.9: Some possible reduction of \( f_2 \) with \( p = 7 \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>subset of ( {x_1, \ldots, x_6} )</th>
<th>( n )</th>
<th>subset of ( {x_1, \ldots, x_6} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>( {x_2, x_6} )</td>
<td>89</td>
<td>( {x_2, x_5, x_6} )</td>
</tr>
<tr>
<td>141</td>
<td>( {x_2, x_4, x_6} )</td>
<td>7</td>
<td>( {x_2, x_5} )</td>
</tr>
<tr>
<td>103</td>
<td>( {x_2, x_4, x_5, x_6} )</td>
<td>8</td>
<td>( {x_2, x_3} )</td>
</tr>
<tr>
<td>32</td>
<td>( {x_2, x_3, x_6} )</td>
<td>2</td>
<td>( {x_2, x_3, x_5} )</td>
</tr>
<tr>
<td>103</td>
<td>( {x_2, x_3, x_5, x_6} )</td>
<td>18</td>
<td>( {x_2, x_3, x_4} )</td>
</tr>
<tr>
<td>259</td>
<td>( {x_2, x_3, x_4, x_6} )</td>
<td>20</td>
<td>( {x_2, x_3, x_4, x_5} )</td>
</tr>
<tr>
<td>174</td>
<td>( {x_2, x_3, x_4, x_5, x_6} )</td>
<td>186</td>
<td>( {x_1, x_2, x_6} )</td>
</tr>
<tr>
<td>14</td>
<td>( {x_1, x_2, x_5} )</td>
<td>178</td>
<td>( {x_1, x_2, x_5, x_6} )</td>
</tr>
<tr>
<td>35</td>
<td>( {x_1, x_2, x_4} )</td>
<td>387</td>
<td>( {x_1, x_2, x_4, x_6} )</td>
</tr>
<tr>
<td>13</td>
<td>( {x_1, x_2, x_4, x_5} )</td>
<td>257</td>
<td>( {x_1, x_2, x_4, x_5, x_6} )</td>
</tr>
<tr>
<td>16</td>
<td>( {x_1, x_2, x_3} )</td>
<td>200</td>
<td>( {x_1, x_2, x_3, x_6} )</td>
</tr>
<tr>
<td>91</td>
<td>( {x_1, x_2, x_3, x_5} )</td>
<td>110</td>
<td>( {x_1, x_2, x_3, x_5, x_6} )</td>
</tr>
<tr>
<td>58</td>
<td>( {x_1, x_2, x_3, x_4} )</td>
<td>520</td>
<td>( {x_1, x_2, x_3, x_4, x_6} )</td>
</tr>
<tr>
<td>80</td>
<td>( {x_1, x_2, x_3, x_4, x_5} )</td>
<td>202</td>
<td>( {x_1, x_2, x_3, x_4, x_5, x_6} )</td>
</tr>
</tbody>
</table>

Table 9.10: Possible dependencies of \( f_2 \) with \( p = 7 \)

\[
\text{decomposition} \quad \begin{array}{c}
\{x_2, x_6\} \\
\{x_2, x_4\} \\
\{x_2, x_3\} \\
\{x_1, x_2\}
\end{array} \quad T_1, S_3
\]

Table 9.11: Primary decomposition for \( g_2 \) with \( p = 7 \)
9.1.1 Comments

In the previous section we clearly see that in every possible step of discretization does not emerge one polynomial among the other. Without further knowledge of the network we cannot give a solid reconstruction of it. The natural question is: the discretization with such little primes could be not a right choice, what happens if we increase the level of the discretization? We try also to increment the discretization size according to the fact that some difference of increasing (decreasing) are deleted for small primes. Moreover we can not use a large prime, in this case probably each time step will have different values in every position (and so the $f_i$ are easily expressible in term of one among $x_1, \ldots, x_6$).

We choose $p = 13$. The discretized data are represented in table 9.13

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>2</td>
<td>9</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 9.12: Discretization with $p = 13$

We obtain 16910 equivalence classes in the Gröbner fan after over one day of computation. The problem is that the calculation of the Gröbner fan is an algorithm very slow. For this reason we want to use primary and double decomposition for the reconstruction of the network.

We will calculate all the possible minimal subsets for all gene $g_i$ using the primary decomposition algorithm (see algorithm 8). Later we will compute also the intrinsic dependencies in order to decrease the number of the possible subsets given by the primary decomposition using the double primary decomposition (see algorithm 9). In table 9.13 we summarize all possible
decompositions for the six genes, and the relative scores.

<table>
<thead>
<tr>
<th>$f_1$</th>
<th>score</th>
<th>$f_2$</th>
<th>score</th>
<th>$f_3$</th>
<th>score</th>
<th>$f_4$</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>${x_4, x_6}$</td>
<td>12</td>
<td>${x_4, x_6}$</td>
<td>12</td>
<td>${x_4, x_6}$</td>
<td>12</td>
<td>${x_5}$</td>
<td>1</td>
</tr>
<tr>
<td>${x_2, x_6}$</td>
<td>12</td>
<td>${x_2, x_6}$</td>
<td>15</td>
<td>${x_2, x_6}$</td>
<td>12</td>
<td>${x_4, x_6}$</td>
<td>12</td>
</tr>
<tr>
<td>${x_1, x_6}$</td>
<td>12</td>
<td>${x_1, x_6}$</td>
<td>15</td>
<td>${x_1, x_6}$</td>
<td>12</td>
<td>${x_3, x_6}$</td>
<td>12</td>
</tr>
<tr>
<td>${x_4, x_5}$</td>
<td>12</td>
<td>${x_2, x_5}$</td>
<td>15</td>
<td>${x_4, x_5}$</td>
<td>12</td>
<td>${x_2, x_6}$</td>
<td>12</td>
</tr>
<tr>
<td>${x_2, x_5}$</td>
<td>12</td>
<td>${x_1, x_5}$</td>
<td>15</td>
<td>${x_2, x_5}$</td>
<td>12</td>
<td>${x_1, x_6}$</td>
<td>12</td>
</tr>
<tr>
<td>${x_1, x_5}$</td>
<td>12</td>
<td>${x_2, x_4}$</td>
<td>20</td>
<td>${x_1, x_5}$</td>
<td>12</td>
<td>${x_2, x_4}$</td>
<td>9</td>
</tr>
<tr>
<td>${x_2, x_4}$</td>
<td>16</td>
<td>${x_1, x_4}$</td>
<td>20</td>
<td>${x_2, x_4}$</td>
<td>16</td>
<td>${x_1, x_4}$</td>
<td>9</td>
</tr>
<tr>
<td>${x_1, x_4}$</td>
<td>16</td>
<td>${x_1, x_3}$</td>
<td>15</td>
<td>${x_1, x_4}$</td>
<td>16</td>
<td>${x_2, x_3}$</td>
<td>9</td>
</tr>
<tr>
<td>${x_1, x_3}$</td>
<td>8</td>
<td>${x_2, x_3}$</td>
<td>15</td>
<td>${x_1, x_3}$</td>
<td>8</td>
<td>${x_1, x_3}$</td>
<td>9</td>
</tr>
<tr>
<td>${x_2, x_3}$</td>
<td>8</td>
<td>${x_1, x_2}$</td>
<td>25</td>
<td>${x_2, x_3}$</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>${x_3, x_4, x_5}$</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$f_5$</th>
<th>score</th>
<th>$f_6$</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>${x_6}$</td>
<td>1</td>
<td>${x_4, x_6}$</td>
<td>12</td>
</tr>
<tr>
<td>${x_4}$</td>
<td>1</td>
<td>${x_2, x_6}$</td>
<td>12</td>
</tr>
<tr>
<td>${x_3}$</td>
<td>1</td>
<td>${x_1, x_6}$</td>
<td>12</td>
</tr>
<tr>
<td>${x_2, x_5}$</td>
<td>2</td>
<td>${x_4, x_5}$</td>
<td>12</td>
</tr>
<tr>
<td>${x_1, x_5}$</td>
<td>2</td>
<td>${x_2, x_5}$</td>
<td>12</td>
</tr>
<tr>
<td>${x_1, x_5}$</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>${x_2, x_4}$</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>${x_1, x_4}$</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>${x_1, x_3}$</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>${x_2, x_3}$</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9.13: Primary decomposition of all genes with $p = 13$

In table 9.13 we underline in red the true dependencies of the network (see figure 9.1). Without extra-knowledge we can not choose which model represent our network, even if we can see that in the pool it is present, with the exception of $g_5$. Looking at the primary decomposition, we note that we can write $f_5$ in function of one of the variables $x_3, x_4$ or $x_5$. This does not yield dependencies. With this level of discretization and so few point, it is quite probable to find just one variable that expresses the function we need. We have to focus to the two other dependences $\{x_1, x_5\}$ and $\{x_2, x_5\}$. These two tell us that there is a strong link between $f_5$ and $x_5$. We can add only an extra link $\frac{1}{5}$ or $\frac{2}{5}$ in order to recreate the link $\frac{5}{5}$.

In table 9.14 we write the results of the double decomposition. In each column we find all the possible minimal subsets of $\{x_1, \ldots, x_n\}$ such that we can express $x_i = h(x_1, \ldots, x_{i-1}, x_{i+1} \ldots, x_n)$. 

\begin{align*}
\{x_4, x_6\} & : 12 \\
\{x_2, x_6\} & : 15 \\
\{x_1, x_6\} & : 15 \\
\{x_4, x_5\} & : 12 \\
\{x_2, x_5\} & : 12 \\
\{x_1, x_5\} & : 12 \\
\{x_1, x_5\} & : 12 \\
\{x_2, x_4\} & : 16 \\
\{x_1, x_4\} & : 16 \\
\{x_2, x_3\} & : 8 \\
\{x_1, x_3\} & : 8 \\
\{x_2, x_3\} & : 8 \\
\{x_3, x_4, x_5\} & : 36 \\
\end{align*}
### 9.1. Gröbner Fan and Primary Decomposition Results

<table>
<thead>
<tr>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$g_3$</th>
<th>$g_4$</th>
<th>$g_5$</th>
<th>$g_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${x_4, x_6}, 6$</td>
<td>${x_3, x_6}, 6$</td>
<td>${x_5, x_6}, 12$</td>
<td>${x_2, x_6}, 6$</td>
<td>${x_4, x_6}, 12$</td>
<td>${x_3, x_5}, 9$</td>
</tr>
<tr>
<td>${x_2, x_6}, 8$</td>
<td>${x_1, x_6}, 6$</td>
<td>${x_4, x_6}, 12$</td>
<td>${x_1, x_6}, 6$</td>
<td>${x_3, x_6}, 12$</td>
<td>${x_2, x_5}, 9$</td>
</tr>
<tr>
<td>${x_2, x_5}, 8$</td>
<td>${x_3, x_5}, 6$</td>
<td>${x_2, x_6}, 12$</td>
<td>${x_2, x_5}, 6$</td>
<td>${x_2, x_6}, 12$</td>
<td>${x_1, x_5}, 9$</td>
</tr>
<tr>
<td>${x_2, x_4}, 12$</td>
<td>${x_1, x_3}, 6$</td>
<td>${x_1, x_6}, 12$</td>
<td>${x_1, x_5}, 6$</td>
<td>${x_1, x_6}, 12$</td>
<td>${x_2, x_4}, 6$</td>
</tr>
<tr>
<td>${x_2, x_3}, 8$</td>
<td>${x_3, x_4}, 6$</td>
<td>${x_1, x_5}, 9$</td>
<td>${x_2, x_3}, 6$</td>
<td>${x_2, x_4}, 9$</td>
<td>${x_1, x_3}, 6$</td>
</tr>
<tr>
<td>${x_3, x_4, x_5}, 12$</td>
<td>${x_1, x_4}, 6$</td>
<td>${x_2, x_5}, 9$</td>
<td>${x_1, x_3}, 6$</td>
<td>${x_2, x_3}, 9$</td>
<td>${x_1, x_3}, 9$</td>
</tr>
<tr>
<td></td>
<td>${x_2, x_4}, 9$</td>
<td></td>
<td>${x_2, x_3}, 9$</td>
<td></td>
<td>${x_1, x_3}, 9$</td>
</tr>
</tbody>
</table>

Table 9.14: Intrinsic dependencies

Using tables 9.14, 9.13 and some knowledge of the network we can reconstruct its topology. We summarize the reconstruction in example 18.

**Example 18.** It is clear that looking only table 9.13 we can not decide how relationship consider. There is not a subset of $\{x_1, \ldots, x_n\}$ that is ”more relevant” of the others. In this case some biological information are indispensable for the reconstruction of the network. Some of those are well-known dependencies, and so it is reasonable to use them in case of a real situation with a real timeseries. Before use some biological information, we try to use the double primary decomposition.

Referring to table 9.14 we can say that $g_1$ at each temporary step depends on $g_2$ and $g_4$ (fourth row, first column). This consideration can arise from the score obtained by this minimal subset. In other cases, this kind of information could be a biological one.

We can use this information, for example, in order to determinate the dependencies of $f_6$. More explicitly, in table 9.13, we can merge the subset $\{x_2, x_4\}$ with the subset $\{x_1, x_4\}$ increasing the score of $\{x_2, x_4\}$ and concluding the analysis of $f_6$. Indeed if we replace $x_1$ with $\{x_2, x_4\}$ in the column of $f_6$, the subset $\{x_1, x_4\}$ becomes $\{x_2, x_4, x_4\} = \{x_2, x_4\}$ and can be merged with the preexisting $\{x_2, x_4\}$. In this way the score of $\{x_2, x_4\}$ rises to 32 and we can say that $f_6 = f_6(x_2, x_4)$. Using this same relation given by the double primary decomposition, we can say that $f_4 = f_4(x_2, x_4)$. Until now we use only the instruments shown in this thesis. For the reconstruction of $f_1$, $f_2$ and $f_3$ we need some extra knowledge of the network. The relationship between $g_1$ and the couple $g_2, g_4$ could traviate us when we apply it in the decomposition of $f_2$. If we use it now we will lost the link $\overrightarrow{3, 2}$ and we will add the link $\overrightarrow{4, 2}$. In this case we need a biological information, for example if we know that $g_2$ depends at least to $g_3$ in the network (not at the same
temporary step but in the network as the link \(3 \rightarrow 2\) we will add only the link \(4 \rightarrow 2\) using the double primary decomposition in the same way of before. In the same way we recovered the dependencies of \(g_2\), we can reconstruct the function \(f_3\). If we know that the link \(5 \rightarrow 3\) exists biologically (in terms of variables this means that \(x_5\) is one of the variables occurring in \(f_3\)), then we have to choose the others variables from the set \(\{x_1, x_2, x_4\}\) (see the third column of table 9.13). One of these three is sufficient for the reconstruction of \(f_3\). Using the primary decomposition as before, we will choose \(x_1\), and so the set \(\{x_1, x_3\}\) for the recovering of \(f_3\). This last choice is made because if we choose \(x_1\) we are really choosing \(x_2\) and \(x_4\) according to the double primary decomposition. In other words, if we choose \(x_1\), that can be expressed by \(\{x_2, x_4\}\), we choose the least restrictive solution. The two subsets \(\{x_2, x_5\}\) and \(\{x_4, x_5\}\) are too subsets of \(\{x_2, x_4, x_5\}\). In the same way we recovered \(f_3\) we can recovered \(f_1\), provided that biologically we know there is the link \(6 \rightarrow 1\).

We can summarize the situation:

1. We looked at the double primary decomposition. In table 9.14 the only one relevant result is that \(x_1 = h(x_2, x_5)\).

2. We use the step 1 directly for the reconstruction of \(f_6\) and \(f_4\).

3. We use some biological information (\(5 \rightarrow 3\), \(6 \rightarrow 1\) and \(3 \rightarrow 2\)) and indirectly the step 1 for the reconstruction of \(f_1\), \(f_2\) and \(f_3\).

4. We use an empirical approach for the reconstruction of \(f_5\).

Finally we can summarize the reverse engineering in figure 9.3 and the confusion matrix in table 9.15.
9.1. GRÖBNER FAN AND PRIMARY DECOMPOSITION RESULTS

Figure 9.3: The left figure represents the network with the three biological informations (green). After the reverse engineering we can recover the original network (green and black) adding two false positive links (blue).

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Negative</td>
</tr>
<tr>
<td>Negative</td>
<td>21</td>
</tr>
<tr>
<td>Positive</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 9.15: Confusion matrix referred to figure 9.3

\[
TP = \frac{d}{c + d} = \frac{13}{0 + 13} = 1,
\]
\[
TN = \frac{a}{a + b} = \frac{21}{21 + 2} \approx 0.913,
\]
\[
FP = \frac{b}{a + b} = \frac{2}{2 + 21} \approx 0.086,
\]
\[
FN = \frac{c}{c + d} = \frac{0}{0 + 13} = 0,
\]
\[
AC = \frac{a + d}{a + b + c + d} = \frac{21 + 13}{21 + 2 + 0 + 13} \approx 0.944.
\]

9.1.2 Conclusions

From the results of the above examples, the introduction of the newly presented strategies can relevantly help the discovery of the functional links among the network nodes. In particular, when coupled with a suitable biological a priori knowledge the whole algorithm is an effective tools in computational biology. Although the computational burden may dramatically increase with the number of nodes (thus making the method almost unfeasible for larger networks), even in the real biological case this procedure can be usefully employed for detecting existing submodules (or motifs) of complex GRN or protein-protein interaction networks. Finally, alternative paths can
be explored as pointed out in the previous chapters which can help easing such computational workload: analysis of these strategies will constitute the natural prosecution of the present work.
Appendix A

Appendix

A.1 Groups, rings and ideals

Definition 52. A group $G$ is a set with a law of composition which is associative, and having a unit element denote by the "$e" (monoid) such that

$$\forall x \in G, \ \exists y \in G \text{ s.t. } xy = yx = e$$

Such an element $y$ is called an inverse for $x$ and it is unique. A commutative group is a group in which $xy = yx \ \forall x, y \in G$.

Definition 53. A ring $A$ is a set, together with two laws of composition called multiplication and addition satisfying the following conditions:

- with respect to addition, $A$ is a commutative group,
- the multiplication is associative, and has a unit element,
- for all $x, y, z \in A$ we have:

$$(x + y)z = xz + yz \quad \text{and} \quad z(x + y) = zx + zy.$$  

This last is the distributive property of multiplication over addition.

Definition 54. A subring $S$ of a ring $A$ is a subset of $A$ closed respect to the operation of $A$ such that $0, 1 \in S$ and results to be a ring.

As usual, we denote the unit element for addition by 0, and the unit element for multiplication by 1. We note that we do not assume that $1 \neq 0$. Standard laws relating addition and multiplication can be found in [Lan71].
Let $A$ be a ring, and let $U$ be the set of elements of $A$ which have both a right and left inverse. Then $U$ is a multiplicative group and it is called the group of **units** of $A$. It is usually denoted by $A^*$ and also called group of **invertible** elements of $A$.

**Definition 55.** A ring $A$ such that $1 \neq 0$, and such that every non-zero element is invertible is called **division ring**. A **commutative** ring $A$, is a ring in which $xy = yx$ for all $x, y \in A$. Moreover, if in a commutative ring $A$ holds that $1 \neq 0$ and there are no zero divisors, then it is called **entire**.

**Definition 56.** A **left ideal** $I$ in a ring $A$ is a subset of $A$ which is a subgroup of the additive group of $A$, and such that $AI \subset I$. To define a right ideal, we required $IA \subset I$. If $I$ is, in the same time, a right and a left ideal, it is called **two-side ideal**. If $A$ is a commutative ring then every left (right) ideal is a two-side ideal. In this case we shall call $I$ simply **ideal**.

From now and for all this thesis the word ring will mean commutative ring.

**Definition 57.** A **prime** ideal $I \neq A$ is an ideal such that, whenever $x, y \in A$ and $xy \in I$, then $x \in I$ or $y \in I$. We say that $I$ is **maximal** if $I \neq A$ and if there is no ideal $I' \neq A$ containing $I$ and $I' \neq I$. Finally, we say that $I$ is **principal** if it is generated by a unique element of $A$.

**Definition 58.** Let $A$ a ring and $I \subset A$ a two-side ideal of $A$. Then the **residue ring** or **quotient ring** $A/I$ is a ring construct by defining an equivalence relation $\sim$ on $A$ as: $a \sim b$ if and only if $a - b \in I$. The set of all equivalence classes is denoted by $A/I$.

**Definition 59.** Let $A$ be a ring, not necessarily commutative. We shall say that $A$ is **Noetherian** if it satisfies any one of the following three conditions:

- every ideal $I$ of $A$ is finitely generated.
- every ascending sequence of ideals is finite.
- every non-empty set of ideals of $A$, partially ordered by inclusion, has a maximal element with respect to set inclusion.

If $A$ is commutative, the first condition can be replaced with:

- every prime ideal $I$ of $A$ is finitely generated.
A.2 Polynomials

We show now a formal approach for the definition of polynomials taken form [Lan71].

Let $S$ a set and $\mathbb{N}$ be the additive monoid of integer $\geq 0$. We denote with $\mathbb{N} \langle S \rangle$ the set of functions $S \to \mathbb{N}$ which are 0 for almost all elements of $S$. If $x \in S$ and $i \in \mathbb{N}$ we denote by $x^i$ the function which takes the value $i$ at $x$ and 0 at $y \neq x$. If $\phi, \psi$ are two functions as above, we define their product $\phi \psi$ as:

$$(\phi \psi)(x) := \phi(x) + \psi(x).$$

It is easy to see that $\mathbb{N} \langle S \rangle$ is a multiplicative monoid in which the unit element is the zero function. Every $\phi \in \mathbb{N} \langle S \rangle$ has a unique expression as a product:

$$\prod_{x \in S} x^{\nu(x)}$$

where $\nu : S \to \mathbb{N}$ is a map such that $\nu(x) = 0$ for almost all $x$. Such a product will be called a primitive monomial, and will be sometimes denoted by $M(\nu)(S)$, or simply $M(\nu)$. We have an injection $j_S : S \to \mathbb{N} \langle S \rangle$ given by $x \mapsto x^1$, whose image generates $\mathbb{N} \langle S \rangle$ as a monoid. We note that if $n$ is a integer $\geq 0$, then

$$(x^1)^n = x^1 x^1 \cdots x^1$$

is equal to $x^n$, i.e. our notation is compatible with the notation used for the product of functions.

All this definition can be used to define polynomials ring.

**Definition 60.** Let $A$ be a ring. We may then form the monoid algebra $A[\mathbb{N} \langle S \rangle]$ over $A$, and we shall call it the polynomial ring (algebra) of $S$ over $A$.

For simplicity, we denote it by $A[S]$. By definition, every element of $A[S]$ has a unique expression as a linear combination

$$\sum_{(\nu)} a_{(\nu)} M(\nu)(S) = \sum_{(\nu)} a_{(\nu)} \prod_{x \in S} x^{\nu(x)},$$

where $(\nu)$ ranges over maps over $S$ into $\mathbb{N}$ for almost all $x$, and $a_{(\nu)}$ is 0 for almost all $(\nu)$. Any polynomial ring in finitely-many variables over a field is Noetherian. This is a consequence of the Hilbert basis theorem ([Lan71], [CLO07]). Elements of $A[S]$ are called polynomials in $S$ over $A$. The elements $a_{(\nu)}$ are called coefficients of the polynomial. If $S$ is a set of $n$ letters $x_1, x_2, \ldots, x_n$, then

$$A[S] = A[x_1, x_2, \ldots, x_n] = A[x],$$
and we call this the polynomial ring (algebra) in $x_1, x_2, \ldots, x_n$ over $A$. Every polynomial of $A[x]$ can be written uniquely as a sum

$$\sum a_{(\nu)}x_1^{\nu_1} \cdots x_n^{\nu_n} \quad (A.1)$$

the sum being taken over all $n$-tuples of integers $\nu_1, \ldots, \nu_n \geq 0$, and almost all coefficients $a_{(\nu)}$ being equal to 0.

**Definition 61.** The degree of a primitive monomial $f = x_1^{\nu_1} \cdots x_n^{\nu_n}$ is defined as $\deg(f) = \nu_1 + \ldots + \nu_n$.

**Definition 62.** A monomial is a polynomial in the form $ax_1^{\nu_1} \cdots x_n^{\nu_n} \ a \in A$ (not necessarily primitive).

If $\alpha(x)$ is a polynomial as A.1 then either $\alpha = 0$, in which case we say that its degree is $-\infty$, or $\alpha \neq 0$, and then we define the degree of $\alpha$ to be the maximum of the degrees of the monomials $M_{(\nu)}(x)$ such that $a_{(\nu)} \neq 0$. (Such monomials are said to occur in the polynomial.) We note that a polynomial in $n$ variables $x_1, \ldots, x_n$ can be viewed as a polynomial in $x_n$ with coefficient in $A[x_1, \ldots, x_{n-1}]$ (if $n \geq 0$). By the degree of $\alpha$ in $x_n$ we shall mean its degree when viewed as a polynomial in $x_n$ with coefficient in $A[x_1, \ldots, x_{n-1}]$. One sees easily that if this degree is $d$, then $d$ is the largest integer occurring as an exponent of $x_n$ in a monomial $ax_1^{\nu_1} \cdots x_n^{\nu_n} \ a \in A$ with $a_{(\nu)} \neq 0$. Similarly, we define the degree of $\alpha$ in each variable $x_i$. The degree of $\alpha$ in each variable is of course usually different from its degree (which is sometimes called the total degree if there is need to prevent ambiguity).

For instance

$$x_1^3x_2 + x_2^2$$

has total degree 4, and has degree 3 in $x_1$ and 2 in $x_2$. As a matter of notation, we shall often abbreviate degree by $\deg$. Let $f(x)$ be a polynomial in one variable in $A[x]$, and write:

$$f(x) = a_0 + \ldots + a_nx^n \ a_i \in A \ n \geq 0.$$

**Definition 63.** If $f \neq 0$, and $\deg f = n$, then $a_n \neq 0$ by definition, and we call $a_n$ the leading coefficient of $f$. And we call $f(0) = a_0$ its constant term.

Let $g(x) \in A[x]$ with $\deg g = m$ and $b_m$ the leading coefficient. Then $f(x)g(x) = a_0b_0 + \ldots + a_nb_mx^{m+n}$. If we assume that at least one of the leading coefficient $a_n$ or $b_m$ is not a divisor of 0 in $A$, then

$$\deg (fg) = \deg f + \deg g$$

and the leading coefficient is $a_nb_m$.

In this situation holds this important theorem:
Theorem 14. Let $A$ be a commutative ring, let $f(x), g(x) \in A[x]$ be polynomials in one variable, of degrees $\geq 0$, and assume that the leading coefficient of $g(x)$ is a unit in $A$. Then there exist unique polynomials $q(x), f(x) \in A[x]$ such that

$$f(x) = g(x)q(x) + r(x)$$

and $\text{deg } r(x) < \text{deg } q(x)$.

Proof. We use induction on $n = \text{deg } f(x)$. Let:

$$f(x) = a_n x^n + \ldots + a_0,$$
$$g(x) = b_d x^d + \ldots + b_0,$$

where $n = \text{deg } f(x)$, $d = \text{deg } g(x)$ and $b_d$ is a unit in $A$.

If $n = 0$, and $d > 0$ then we let $q(x) = 0$ and $r(x) = f(x)$. If $\text{deg } g(x) = 0$ we let $r(x) = 0$ and $q(x) = a_0 b_0^{-1}$.

Assume now the theorem proved for polynomials of degree $< n$. We may assume that $d \leq n$ (otherwise, take $q(x) = 0$ and $r(x) = f(x)$).

Then:

$$f(x) = a_n b_d^{-1} x^{n-d} g(x) + h(x)$$

where $\text{deg } h(x) < n$. By induction, we can find $q_1(x)$ and $r(x)$ such that:

$$f(x) = a_n b_d^{-1} x^{n-d} g(x) + q_1(x) g(x) + r(x)$$

and $\text{deg } r(x) < d$. Then we let

$$q(x) = a_n b_d^{-1} x^{n-d} + q_1(x)$$

and we conclude the proof of existence. For the uniqueness, we suppose, according to hypothesis:

$$f(x) = q_1(x) g(x) + r_1(x) = q_2(x) g(x) + r_2(x).$$

Subtracting we obtain:

$$(q_1(x) - q_2(x)) g(x) = r_2(x) - r_1(x).$$

Since $b_d$ is assumed to be a unit we have:

$$\text{deg }((q_1(x) - q_2(x)) g(x)) = \text{deg } (q_1(x) - q_2(x) + \text{deg } g(x)).$$

Since $\text{deg } (r_2(x) - r_1(x)) < \text{deg } g(x)$, the previous relation can hold only if $q_1(x) - q_2(x) = 0$ and the proof is completed. \qed
There are some important properties about polynomial rings ([Lan71], [CLO07]):

- any polynomial ring in finitely-many variables over a Noetherian ring is Noetherian. This is a consequence of the Hilbert basis theorem.

- Any polynomial ring in finitely-many variables over integral domain ring is an integral domain.

- Any polynomial ring in finitely-many variables over a unique factorization domain is an unique factorization domain. The proof is based on the Gauss lemma.

**Definition 64.** A field is a division commutative ring.

About fields we will speak more in this section.

### A.3 Finite fields

As we say before a field $K$ is a commutative ring with unit in which every not 0 element has an inverse. Let $A$ a ring and suppose that $1 \in A$. Let us find the smallest non banal subring of $A$ containing 1. If $P$ is this subring, then $P$ contains $\{1, 2, 3, \ldots \}$ and also $\{-1, -2, \ldots \}$. And so $P$ contains $Q = \{m \cdot 1 \in \mathbb{Z}\}$. It is easy to see that $Q$ is a subring of $A$.

Let we now consider

\[
\phi : \mathbb{Z} \to A \\
m \mapsto m \cdot 1
\]

It is clear that $\phi(\mathbb{Z}) = Q$ and that $\phi$ is a morphism. Now, if $\phi$ is injective than multiples of 1 are distinct and then $\phi : \mathbb{Z} \to Q$ is a bijection and so an isomorphism.

If $\phi$ is not injective then

\[
\exists m, n \in \mathbb{Z} \ m \neq n \text{ s.t. } \phi(m) = \phi(n) \Rightarrow m \cdot 1 = n \cdot 1
\]

suppose $m > n$

\[
(m - n) \cdot 1 = 0
\]

and call $K = \{k \in \mathbb{Z} : k > 0, k \cdot 1 = 0\}$,

now $K \neq \emptyset$ and so for the well-ordering principle has a minimum that we call $t$. 
Definition 65. In the previous conditions \( t \) is called additive order or characteristic.

We write now \( a|b \) with \( a \neq 0 \) if \( a \) divides \( b \).

Lemma 11. Let \( n \in \mathbb{Z} \) then, \( n \cdot 1 = 0 \iff t|n \).

Proof. \( \Leftarrow \): if \( t|n \) then \( n = t \cdot q \) with \( q \in \mathbb{Z} \). And so \( n \cdot 1 = (t \cdot q) \cdot 1 = (t \cdot 1) \cdot (q \cdot 1) = 0 \).

\( \Rightarrow \) using euclidean algorithm we write \( n = t \cdot q + r \) with \( 0 \leq r < t \). And so \( 0 = n \cdot 1 = (t \cdot q) \cdot 1 + r \cdot 1 = r \cdot 1 \). But \( r < t \) and \( t \) is minimal and so \( r = 0 \). \( \Box \)

Finally

\[ m \cdot 1 = n \cdot 1 \iff t|(m - n) \iff m \equiv n \text{ mod } t \iff [m] = [n] \in \mathbb{Z}_t. \]

Let consider now

\[ \psi : \mathbb{Z}_t \to A \]
\[ [n] \mapsto n \cdot 1 \]

For the consideration just seen, this map is good defined and so injective, and this implies a bijection between \( \mathbb{Z}_t \) and \( Q = \psi(\mathbb{Z}_t) \).

In conclusion, if a ring \( A \) has characteristic \( t \) (has additive period of unit equal to \( t \)) then \( A \supset Q \simeq \mathbb{Z}_t \). We have that \( \mathbb{Z}_t \) is a domain if and only if \( t \) is prime. In this case it is also a field. If \( A \) is a domain then \( t \) is prime. Finally, if \( A \) is a finite field then its characteristic is \( p \) a prime and so \( A \) contains a isomorphic copy of \( \mathbb{Z}_p = \mathbb{F}_p \).

Definition 66. A subfield \( F \) of a field \( E \), is a subset of \( E \) closed by the operation of \( E \) and results to be field. We say also that \( E \) is a extension of \( F \). In this situation, an element \( \alpha \) of \( E \) is said algebraic over \( F \) if there exist \( a_0, \ldots a_n \in F \), not all equal to 0 and \( n \geq 1 \) such that:

\[ a_n \alpha^n + \ldots + a_0 = 0 \]

There is an other equivalent definition. Let \( x \) a variable over \( F \) and let consider the homomorphism \( F[x] \to E \) that is the identity over \( F \) and maps \( x \) on \( \alpha \) has a non-zero kernel. This definition is useful because the non-zero kernel of the isomorphism is a principal ideal, and so generated by only one element \( p(x) \) of \( F[x] \) that we can suppose monic. For the ring and ideal theory, we have that \( F[x]/(p(x)) \) is isomorphic to \( F[\alpha] \) and since \( F[\alpha] \) is entire, it follows that \( p(x) \) is irreducible. For the fact that \( p(x) \) is monic, this is the irreducible polynomial of \( \alpha \) over \( F \).
Definition 67. An extension $E$ of $F$ is called algebraic if every element of $E$ is algebraic over $F$.

We can see $E$ a vector space over the field $F$. In this way we say that the extension is finite if the dimension of the vector space is finite, otherwise it is called infinite extension.

**Lemma 12.** Let $E$ a finite extension of $F$. Then $E$ is algebraic over $F$.

The viceversa is not always true: the subfield of the complex numbers consisting of all algebraic numbers over $Q$ is an infinite extension of $Q$.

A finite field $E$ contains a copy of $F_p$, the field with $p$ element (with $p$ prime). We can see $E$ as a vectorial space over $F_p$ of dimension $n = |E : F_p| < +\infty$. It is easy to see that in this way a finite field has $p^n$ elements. Moreover, for all $p$ prime and $n > 0$ exist a field with $p^n$ elements. We use a heuristic approach: if $E$ is a field with $p^n$ elements then $E^*$ has $p^n - 1$ elements and results to be a multiplicative group. For the Lagrange’s theorem we have that

$$\forall a \in E^* \quad a^{|E^*|} = a^{p^n - 1} = 1.$$  

Then $a^{p^n} = a$ and so the elements of $E$ are the solution of

$$x^{p^n} - x \in F_p[x].$$

The heuristic approach hints to consider the polynomial $f(x) = x^{p^n} - x \in F_p[x]$. Let $K$ its splitting field over $F_p$ (the smallest extension that contains all roots of $f$). This splitting field is unique at least of isomorphism that fix $F_p$. Let $E = \{ \alpha \in K : f(\alpha) = 0 \}$. We note that $|E| \leq p^n$ indeed, for the fundamental theorem of algebra, the polynomial $f$ has at most $p^n$ solutions. Moreover the roots are distinct. If $\alpha$ is a multiple root, then $\alpha$ is a root of the formal derivative of $f^1$. But the derivative of $f$ is $f' = p^n x^{p^n - 1} - 1 = -1$ thanks to the characteristic of $E$. But $-1$ does not have roots, and so $f$ does not have multiple roots. For this reason $E$ has exactly $p^n$ elements. Indeed $E$ is a subring of $K$, it is obvious that $\alpha, \beta \in E$ implies that $\alpha \beta \in E$. For the sum we note that $(\alpha + \beta)^{p^n} = \alpha^{p^n} + \beta^{p^n}$ because $p|i$ if $p$ is prime and $0 < i < p$. Now, we have a subring in a field, and so $E$ is a domain because $K$ it is and for Wedderburn’s little theorem [Par83] $E$ is a field and $E = K$.

In conclusion $F_{p^n}$ is the splitting field over $F_p$ of $f(x) = x^{p^n} - x$, and it is the set of the roots of $f$.

We use the follow theorem (without the proof) for the construction of $F_{p^n}$.

---

1Formal derivative means standard derivative for polynomials $(x^a)' = ax^{a-1}$
Theorem 15. Let $K$ a field and $G$ a subgroup of $K^*$. If $G$ is finite then $G$ is cyclic.

Definition 68. A polynomial $f \in K[x]$ is called **irreducible** if it is non-constant and cannot be represented as the product of two or more non-constant polynomials from $K[x]$.

Let $E$ a finite field. Then $E^*$ is a finite group of order $p^n - 1 = q - 1$ and it is a subgroup of itself, and so for the previous theorem $E^*$ is cyclic. Let $\alpha \in E^*$ of order $q - 1$ and so $E^* = \{0, 1, \ldots, \alpha^{q-2}\}$. Let now consider $F_p[\alpha]$, the smallest subring of $E$ that contains $F_p$ and $\alpha$. If $\alpha \in F_p[\alpha]$, then all power of $\alpha$ are in $F_p[\alpha]$ and so $E = F_p[\alpha]$ moreover $E$ is a simplex extension of $F_p$. The finite fields’ theory assure us that exists a polynomial $f$ such that:

- $f(\alpha) = 0$,
- $f$ is monic,
- $\deg(f)$ is the smallest between all polynomials in $F_p[x]$ satisfying these properties.

For the theory of extension the degree of $f$ is the same of the degree of the extension $|E : F_p| = n$. In conclusion we have prescription for the construction of a finite field of $p^n$ elements:

- Let $F_p = \{0, 1, \ldots, p-1\}$.
- Choose $f \in F_p[x]$, monic, irreducible of degree $n$.
- Let $\alpha$ root of $f$ and $E = F_p[\alpha]^2$.

Example 19. We want to find a finite field of 8 elements:

- $8 = 2^3$, and so we consider $F_2 = \{0, 1\}$.
- Choose $f \in F_2[x]$, monic (gratis) , irreducible of degree 3. Since the degree is $\leq 3$ irreducible means without roots in the base field$^3$. And so $f = x^3 + ax^2 + bx + 1$, where $a, b \in F_2$. We note that there is always the constant term ( to avoid the 0-root). Now, if $a = b$ then 1 is a root, and so we can choose $a = 1, b = 0$ or viceversa. We take both polynomials $f = x^3 + x + 1$ and $g = x^3 + x^2 + 1$.

$^2$This representation is only available when $f$ is a primitive polynomial, for details see after the example.

$^3$for degree $>3$ this condition is not sufficient.
Let $\alpha$ root of $f$ and $\beta$ a root of $g$. And so we have that $\alpha^3 + \alpha + 1 = 0$ and $\beta^3 + \beta^2 + 1 = 0$ and we will use the relations $\alpha^3 = \alpha + 1$ (the sign $-$ in $F_2$ is the same as $+$ since the characteristic is 2), and $\beta^3 = \beta^2 + 1$.

Let now use the fact that finite fields are cyclic (as seen above) and built the two extensions:

$E_1$                      $E_2$
---                       ---
$\alpha^0 = 1$             $\beta^0 = 1$
$\alpha^1 = \alpha$        $\beta^1 = \beta$
$\alpha^2 = \alpha^2$      $\beta^2 = \beta^2$
$\alpha^3 = \alpha + 1$    $\beta^3 = \beta^2 + 1$
$\alpha^4 = (\alpha + 1)\alpha = \alpha^2 + \alpha$  $\beta^4 = \beta^2 + \beta + 1$
$\alpha^5 = (\alpha^2 + \alpha)\alpha = \alpha^3 + \alpha^2 + \alpha^2 + \alpha + 1$  $\beta^5 = \beta + 1$
$\alpha^6 = (\alpha^3)^2 = (\alpha + 1)^2 = \alpha^4 + 1$  $\beta^6 = \beta^2 + \beta$
$\alpha^7 = (\alpha^2 + 1)\alpha = \alpha^3 + \alpha = \alpha + 1 + \alpha = 1$  $\beta^7 = 1$

These two extensions are both $F_{2^3}$ and so these two representation are isomorphic. It is easy to see that $\alpha = \beta + 1$ and that the roots of $f$ are the inverse of the roots of $g$ and viceversa.

If we try to represent $F_{2^4}$ we need an irreducible polynomial with boolean coefficient of degree 4. It is easy to see that $f(x) = x^4 + x^3 + x^2 + x + 1$ is irreducible. If $\alpha$ is a root of $f(x)$ we have that $\alpha^4 = \alpha^3 + \alpha^2 + \alpha + 1$ and so $\alpha^5 = 1$. This only means that with $\alpha$ root of $f$ we can not represent $F_{2^4}$. Without give more details, the finite field $F_{2^4}$ can be viewed as a isomorphic copy of $F_2[x]/\langle f \rangle$. To avoid this situation and use a root of $f$ for the representation of the field, we can use a special polynomial, the Conway polynomial [HL99] or, more generally a primitive polynomial. In this case we can always describe the field using a root of the polynomial.

**Definition 69.** A polynomial $f \in F_p[x]$ is called primitive if $f$ has a root of multiplicative period equals to $p^n - 1$ where $n$ is the degree of $f$.

**Definition 70.** The Euler’s totient function is a function $\varphi : N \to N$ is defined to be the number of positive integers less than or equal to $n$ that are coprime to $n$.

There are a lot of properties of this fundamental function, but they are not the aim of this work. This function is useful to count the number of primitive polynomial, that are $\frac{\varphi(p^n - 1)}{n}$. The proof of this can be found using Galois theory.

We end this section with this fundamental result:

**Theorem 16.** Let $E$ a finite field of characteristic $p$. Let $f : E \to E$ to be a function. Then $f$ is a polynomial in $E[x]$.
Proof. Let \( f, g \in \mathbb{F}_p[x] \). Suppose that \( f(a) = g(a) \) for all \( a \in \mathbb{F}_p \).
Then \( (f - g)(a) = f(a) - g(a) = 0 \). For the Ruffini’s theorem we have that:
\[
x - a | f - g \quad \forall a \in \mathbb{F}_p \iff LCM_{a \in \mathbb{F}_p}(x - a) | f - g.
\]
This lasts for the properties of \( LCM \). The \( LCM_{a \in \mathbb{F}_p} \) is the product of all the factors since the elements of \( \mathbb{F}_p \) are distinct. But this product is equal to \( x^{p^n} - x \) as we see before. And so we have:
\[
LCM_{a \in \mathbb{F}_p}(x - a) | f - g \iff x^{p^n} - x | f - g \iff f \equiv g \mod x^{p^n} - x
\]
This means that a polynomial in \( \mathbb{F}[x] \) has the form \( f(x) = a_0 + a_1 x + \ldots + a_{p^n-1} x^{p^n-1} \) with \( a_i \in \mathbb{F} \). Let consider now \( |E|^E \) the number of functions \( f : E \to E \). But \( |E|^E = |E|^{|E|} = |E|^{p^n} \) which is the numbers of polynomials in \( \mathbb{F}[x] \) because we have to choose the \( p^n \) coefficients of \( f(x) \) in \( \mathbb{F} \). Finally we have so many different polynomials as the function and the proof is complete.
\( \square \)
Bibliography


[bio] [http://www.biochemweb.org/genes.shtml](http://www.biochemweb.org/genes.shtml).


