Polynomial Models for Systems Biology: Data Discretization and Term Order Effect on Dynamics

Elena S. Dimitrova

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mathematics

Reinhard C. Laubenbacher, Chair
Christopher A. Beattie
John A. Burns
Pedro J. Mendes

August 1, 2006
Blacksburg, Virginia

Keywords: Discrete Modeling, Data Discretization, Finite Fields, Polynomial Rings, Polynomial Dynamical Systems, Gröbner Bases, Systems Biology, Biochemical Networks

Polynomial Models for Systems Biology: Data Discretization and Term Order Effect on Dynamics

Elena S. Dimitrova

ABSTRACT

Systems biology aims at system-level understanding of biological systems, in particular cellular networks. The milestones of this understanding are knowledge of the structure of the system, understanding of its dynamics, effective control methods, and powerful prediction capability. The complexity of biological systems makes it inevitable to consider mathematical modeling in order to achieve these goals.

The enormous accumulation of experimental data representing the activities of the living cell has triggered an increasing interest in the reverse engineering of biological networks from data. In particular, construction of discrete models for reverse engineering of biological networks is receiving attention, with the goal of providing a coarse-grained description of such networks. In this dissertation we consider the modeling framework of polynomial dynamical systems over finite fields constructed from experimental data. We present and propose solutions to two problems inherent in this modeling method: the necessity of appropriate discretization of the data and the selection of a particular polynomial model from the set of all models that fit the data.

Data discretization, also known as binning, is a crucial issue for the construction of discrete models of biological networks. Experimental data are however usually continuous, or, at least, represented by computer floating point numbers. A major challenge in discretizing biological data, such as those collected through microarray experiments, is the typically small samples size. Many methods for discretization are not applicable due to the insufficient amount of data. The method proposed in this work is a first attempt to develop a discretization tool that takes into consideration the issues and limitations that are inherent in short data time courses. Our focus is on the two characteristics that any discretization method should possess

in order to be used for dynamic modeling: preservation of dynamics and information content and inhibition of noise. Given a set of data points, of particular importance in the construction of polynomial models for the reverse engineering of biological networks is the collection of all polynomials that vanish on this set of points, the so-called ideal of points. Polynomial ideals can be represented through a special finite generating set, known as Gröbner basis, that possesses some desirable properties. For a given ideal, however, the Gröbner basis may not be unique since its computation depends on the choice of leading terms for the multivariate polynomials in the ideal. The correspondence between data points and uniqueness of Gröbner bases is studied in this dissertation. More specifically, an algorithm is developed for finding all minimal sets of points that, added to the given set, have a corresponding ideal of points with a unique Gröbner basis. This question is of interest in itself but the main motivation for studying it was its relevance to the construction of polynomial dynamical systems.

This research has been partially supported by NIH Grant Nr. RO1GM068947-01.

Dedication

To my parents, for their unending love and support.

Acknowledgements

I first offer my gratitude to my advisor, Reinhard Laubenbacher, for his mentoring and friendship. His tireless commitment to his work has been an incredible source of inspiration for me. I am thankful for the opportunity to be a student of his and for the continuous patience and encouragement I have been receiving. In my future work as a researcher and student advisor, my hope is to come close to the extremely high standards he has established for me.

I thank the members of my committee, Christopher Beattie, John Burns, and Pedro Mendes, for their insightful comments and questions about my work.

I give special thanks to my friend and mentor, Abdul Salaam Jarrah, for the time he dedicated to patiently guiding me through my work. Over the years I have known him, I have come to greatly appreciate his brilliant mind and generous nature. I doubt I will ever be able to repay him for all the help he has given me.

I give warm thanks to the current and former faculty, staff, and graduate students at the Virginia Bioinformatics Institute for providing an excellent interdisciplinary working environment that has determined the course of my future career. I especially thank Diogo Camacho, Autumn Clapp, Miguel Colón-Vélez, Alberto de la Fuente, Edgar Delgado-Eckert, Ina Hoeschelle, Stefan Hoops, John McGee, Ana Martins, Bharat Mehrotra, Pedro Mendes, Dustin Potter, Wei Sha, Vladimir Shulaev, Brandy Stigler, Leepika Tuli, Alan Veliz-Cuba, Paola Vera Licona, Jim Walke, and Dedra Wright. I also thank Michael Stillman from Cornell University for his valuable contribution to several research projects I have been part of.

I thank the professors at Virginia Tech and the American University in Bulgaria who gave me the education and courage to pursue a doctoral degree in mathematics. I am especially thankful to my undergraduate advisor, Oleg Yordanov, without whom I would not have become a mathematician. I also thank Martin Day, Alexander Ganchev, Peter Haskell, Peter Linnell, Stoyan Nedev, and Charles Parry for being incredible teachers and mentors. I am grateful to William Greenberg for helping me come to Virginia Tech.

Next, I give thank to my family, for without their love and support I would not be here. I thank my parents who have thought me to believe in

myself, trusted my choices, and always been there for me whenever I was in need. I thank my brother who had to grow up with his sister far away but kept his heart open for me. I could not have asked for a more loving and supportive family. I honor the memory of my grandfathers whose lives and work never cease to inspire me.

Finally, I thank my friends, scattered all over the world, whose presence in my life has made me the person I am now. I keep dear in my heart my best friend, Nevena Hranova, for her unconditional love and understanding. I thank Hardus Odendaal, who was so patient and supportive during the last stage of my graduate studies, for the happiness he has given me. I am grateful for the friendship of Katarina and Emery Conrad, Nikolay Kolev, Katerina Kormousheva, Maria Laubenbacher, Samantha Lowell, Sequan, Elitza Stanoeva, and Svetla Todorova-Zlatkova.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Abstractness, sometimes hurled as a reproach at mathematics, is its chief glory and its surest title to practical usefulness.

E.T. Bell

The current work does not propose a model of a biological system, nor does it introduce a new modeling method, and in general the theory and algorithms it presents can be used outside of any modeling framework. However, it was clearly motivated by the desire to study the structure and dynamics of biological systems using the techniques of mathematical modeling. We believe that we have contributed to the improvement of the set of tools available to the modeler and therefore feel the need to share our faith in the value of mathematical modeling in biology, and especially in systems biology.

Technological advances in the life sciences have triggered an enormous accumulation of experimental data representing the activities of the living cell. This in turn caused a paradigm shift in biology, resulting in its "moving from being a descriptive science to being a quantitative science" [1]. The emergence of systems biology is part of this paradigm shift. Its goal to study entire biological systems through perturbing them on the molecular level and accurately measuring their response can only be realized with the help of precise and powerful technology. Thus, the movement of biology to quantitative

science opens a two-way street: technology causes quantitative development in biology but the resulting development in the biological sciences requires continued technological developments and increases the need for improved communication between biologists and mathematicians.

1.1 The "System Level" of Systems Biology

Biology is the science of life: it is concerned with the characteristics and behavior of organisms, their origin, and how they interact with each other and the environment. This enormously broad scope has naturally created different levels at which biology studies life – from the atomic and molecular scale of biochemistry and molecular genetics to the level of entire populations in population biology.

The study of life has become an interdisciplinary enterprise in which many academic fields participate by applying their already existing tools and creating new ones to study various biological objects. This way, biology has given rise and become part of fields like biophysics, biomathematics, and bioinformatics. When in the 1950s L. von Bertalanffy, W. R. Ashby, and others [2] founded the field called systems theory, it was only natural that it eventually found its application in the development of *systems biology*. For various reasons this did not happen quickly, though. Over a half-century ago, N. Wiener suggested that living organisms be viewed as systems governed by feedback control [3]. Wiener attempted to found a new discipline – "cybernetics" – for the study of such systems. His ideas generated some excitement in the social sciences in the 1950s, only to fade away soon after, although not disappear completely. At the beginning of the twenty-first century, Wiener's vision has returned, thanks to a great extend to the work of Hiroaki Kitano [4].

Today, only a few years later, Systems Biology is an emergent field that has been ascribed various definitions due to its extremely dynamic development and the large scope of topics it encompasses. As the name suggests, the field aims at *system-level* understanding of biological systems. What does it mean to understand at the "system level"? First, it has to be stipulated that in the present work the focus will be on biological systems at the cell level, without claiming that it is disconnected from the other levels of organization. Unlike molecular biology which focuses on molecules, such as sequences of nucleotide acids and proteins, systems biology's focus is on systems that

are composed of molecular components. The essence of a system lies in its dynamics that cannot be described merely by enumerating the components of the system. Therefore, systems biology performs a shift from molecular characterization to understanding of the functional activity of a biological system. This is not to give support to the claims that the so-called reductionist approach (the detailed study of individual elements of the system) to biology must always fail, either because of nature's redundancy and complexity, or because we have not understood all the parts of the processes. It is probably misleading to believe that only system structure, such as network topology, is important without paying sufficient attention to the diversity and functionality of the components. Both the structure of the system and its components play indispensable role in forming the dynamics of the system as a whole. But, while biologists have always known that a protein must function within the context of the whole cell, it has only recently become possible to obtain data about this functional level [5].

1.2 Role of Mathematical Modeling in Systems Biology

As one moves from the "micro-" to the "macro-" scale of biology, it is interesting to notice that mathematics has increasingly well-established applications, with population biology being one of the oldest area in mathematical biology. On the other end of the spectrum is molecular and cell biology, to which mathematics has found application only a few decades ago. This (inevitable) collaboration is greatly due to the advances in technology that created high-throughput measurements that require computational and analytical tools in order to be utilized. An example of the impact of mathematics on molecular biology is the sequencing of the three billion base pair human genome, achieved through sequencing hundreds of thousands of smaller randomly overlapping contiguous genetic segments [6]. In order to arrange these segments in the biologically correct linear order, one must compute the overlap probability among segments. After having assembled the bits of the genome and obtaining the correct linear sequence, one must then find the genes, the short, scattered regions of DNA which code for currently unknown proteins. This is a highly complicated mathematical problem of

pattern recognition and biological cryptology that earned mathematics an undeniably important role in the post-genomic era of biology.

Perhaps the most significant application of mathematics in biology is the mathematical modeling of biological processes. To the biologist, mathematical modeling offers a research tool, "commensurate with a new powerful laboratory technique," if it is used appropriately and with its limitations recognized [7]. In systems biology, using knowledge from molecular biology, one can propose hypotheses that explain a system's behavior and these can be used to mathematically model the system. Models are then used to predict how different changes in the system's environment affect the system and can be iteratively tested for their validity [8].

To the mathematician, biology is the next great challenge that is going to be a continuous source of motivation and advancement, similar to the way mathematics has and will benefit from its involvement with physics [9]. It is very likely that biology will stimulate the creation and development of entirely new areas of mathematics and contribute to the progress of the already exiting ones. In this process of mutual benefit between biology and mathematics, systems biology plays a significant role. One of the main reasons is its multivariate nature. "You have to be looking at multiple variables simultaneously and how they interact with one another, rather than any specific single variable in isolation," Douglas A. Lauffenburger, professor of biological engineering at Massachusetts Institute of Technology, says about systems biology and establishes mathematical modeling as a natural tool to fulfill this requirement. Another reason for the need of modeling in systems biology is that the wide range of scales on which systems biology operates – from one pathway to a collection of pathways, from one cell to many cells interacting – requires a high level of generalization. Mathematical models provide generalization through abstraction – a gene regulatory network is described in terms of equations. If differential equations are used, for example, their structure and the values of the parameters establish a relationship to a particular gene network. One of the values of the mathematical models is that their formal study allows us to investigate generic properties and test hypotheses. Perhaps equally importantly, the modeling process itself "teaches" us a lot about the functional activity of the system and the dynamic interactions of its components.

1.3 Mathematical Modeling and the *S. cerevisiae* Oxidative Stress Project

This systems biology project is a collaboration between the Laubenbacher, Mendes, and Shulaev labs at the Virginia Bioinformatics Institute. It is funded by the NIGMS under grant number RO1 GM068947-01 and has partially supported the current dissertation. It is briefly presented here as an example of a mathematics-systems biology project and as being a major source of motivation for the work presented in the subsequent chapters.

1.3.1 Biological Motivation

The yeast *Saccharomyces cerevisiae*, also called brewer's or baker's yeast, is a single cell eukaryotic microorganism. It is one of the simplest eukaryotes but many essential cellular processes are conserved between yeast and human. Many yeast proteins have been shown to be functionally interchangeable with their homologous human proteins. Yeast is easy to culture and manipulate genetically and biochemically, and there is vast knowledge for this organism at the physiological, genetic, and molecular levels. Consequently, *S. cerevisiae* is widely used as a model system to understand the molecular mechanisms of oxidative stress response [10].

Oxidative stress is a harmful condition in a cell, tissue or organ caused by reactive oxygen species (ROS) or other molecules with high oxidative potential. All aerobic organisms are exposed to ROS. ROS such as the superoxide anion, H_2O_2, and the hydroxyl radical are generated in the course of normal aerobic metabolism or after exposure to radical-generating compounds. ROS can cause wide-ranging damage to macromolecules, including proteins, lipids, DNA and carbohydrates. ROS have been recognized as important pathophysiologic components of a number of diseases, such as Alzheimer's disease [11], diabetes [12], and cancers [13]. To protect against oxidant damage, cells contain effective defense mechanisms including enzymes and low-molecular-weight compounds [14]. Oxidative stress occurs when the formation of oxidants exceeds the ability of antioxidant systems to remove them. These ROS and the corresponding cellular defense systems have been studied extensively in *S. cerevisiae* [15].

This project studies the kinetics of the *S. cerevisiae* response to oxidative

stress induced by cumene hydroperoxide. Since one of the main objectives of the project was to create mathematical models of biological systems, the two modeling teams, Laubenbacher and Mendes research groups, participated in the experimental design. This was to assure that the data collected would be of the appropriate type, amount, and format for the models of the oxidative stress response system that they were going to build.

1.3.2 Top-down Mathematical Modeling of Oxidative Stress Response

The goal of the project is to develop new techniques to construct integrative mathematical models of biological systems and focuses on the yeast *Saccharomyces cerevisiae* oxidative stress response as an example. The two research teams working on it use different modeling frameworks but have both adopted the so-called *top-down* or *reverse engineering* method for model construction. This method builds a model from observation of the system in response to specially designed perturbations. In this project the observations are measurements of concentrations of biomolecules recorded at predetermined time intervals, resulting in time courses of experimental data. The starting point of the modeling process is the system (the observations) and the result is a model.

The Laubenbacher group has developed methods to create top-down models of biological systems using **discrete mathematics**. The group has developed a new modeling approach for gene regulatory networks from DNA microarray data, based on the framework of discrete dynamical systems in which each variable can take on a finite number of states. Using techniques from computational algebra, a method to reverse-engineer biochemical networks has been constructed. It is introduced in [47] and is also discussed in Section 3.3. The current dissertation presents results that were motivated by the requirements of this modeling approach and can be considered part of it.

The Mendes group has developed methods based on **(continuous) nonlinear dynamics** to create top-down models of biological systems. They use ordinary differential equations for the modeling of biochemical networks as continuous systems. This approach is based on chemical kinetics theory [16].

Chapter 2

Discrete Models for Systems Biology

As discussed in the previous section, mathematics is becoming a key player in the emerging field of systems biology. In the long history of collaboration between biology and mathematics, systems of ordinary differential equations (ODE) have been established as the most common modeling framework. This is due to the well developed mathematical theory behind ODE models and certainly to the large number of biological modeling projects completed successfully using ODEs. A well know example is the Michaelis-Menten ODEs from 1913 that describe the kinetics relationship between substrate concentration and enzyme concentration when the concentration of enzyme is much less than the concentration of the substrate (see [17] for a current derivation).

Mathematical modeling in systems biology is no exception. For gene regulatory networks, for instance, the most common approach to the modeling of dynamics is to view a gene regulatory network as a biochemical network of gene products, typically mRNA and proteins, and to describe their rates of change through a system of ODEs. An example is [18].

Then why do we claim that discrete models are necessary and useful in biology? Inside cells, biochemical reactions are at the lowest level discrete events in which individual molecules and enzymes are brought together for oxidation, reduction, hydrolysis, catalysis, *etc.* [19]. Given current measurement technology, however, it is impractical to measure whole-genome expression levels at single-molecule resolution. For this reason, large numbers of cells are pooled together and mRNA is removed from the population as a

whole. Nevertheless, due to the small amount of data about gene regulatory networks, there are good reasons to choose variable discretization into a small number of levels. There is increasing evidence that certain types of gene regulatory networks have key features that can be represented well through discrete models [20]. For example, it seems reasonable that for many genes, transcription occurs in one of a small number of states, perhaps low/high, off/low/high, low/medium/high, off/low/medium/high, *etc.*, and that the level of transcription of a gene does not smoothly interpolate between these states but rather these states approximate transcriptional equilibria maintained by the cell through its gene regulatory network [19].

But, as observed in [47], continuous and discrete modeling approaches might not be as far apart as it appears. It is useful to keep in mind that most ODE models cannot be solved analytically and that numerical solutions of such systems involve the approximation of the time-continuous system by a time-discrete one. Furthermore, when validating an ODE model using microarray data it is often necessary to utilize thresholds for continuous concentrations. The connection between an ODE system and an associated discrete system that captures information about the continuous dynamics has been formalized in [21].

Next we present briefly several discrete modeling frameworks that have found application in systems biology. In Section 3 we give a more detailed description of another one, polynomial models over finite fields.

2.1 Boolean Networks

Boolean networks were first used in the life sciences in the 1960s, when Stuart Kauffman introduced them to model regulatory networks as switching networks [53]. Boolean network models have the advantage of being more intuitive than ODE models and might be considered as a coarse-grained approximation to the network. They differ from ODE models in that time is taken as discrete and gene expression is discretized into only two quantitative states, as either present or absent.

A Boolean network $G(V, F)$ is a set $V = \{v_1, \ldots, v_n\}$ of vertices representing nodes of the network, together with a collection $F = (f_1, \ldots, f_n)$ of Boolean functions assigned to each node. A Boolean function is a function of the form $f : B^k \to B$ where $B = \{0, 1\}$. The simulation of the transition of

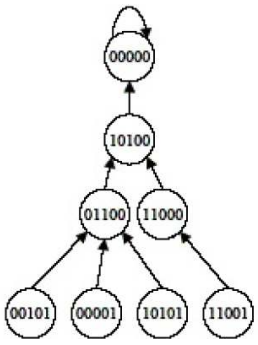


Figure 2.1: A state-transition diagram for a Boolean network.

the system from state to state is represented in the form of a directed graph like the one on Fig. 2.1.

Each v_i represents the state (expression) of gene i, where $v_i = 1$ represents the fact that gene i is expressed and $v_i = 0$ means it is not expressed. The list of Boolean functions F represents the rules of regulatory interactions among the genes. That is, any given gene transforms its inputs (regulatory factors that bind to it) into an output, which is the state or expression of the gene itself. All genes are assumed to update synchronously in accordance with the functions assigned to them and this process is then repeated. The artificial synchrony simplifies computation while preserving the qualitative, generic properties of global network dynamics [22].

2.2 Bayesian Networks

Bayesian networks are a type of probabilistic graphical models. They represent joint probability distribution of a set of variables with explicit independency assumptions. Bayesian networks are used for, among other applications, inferring casual dependencies between genes in gene regulatory networks with the goal of estimating the posterior probability of chosen features being inherent in the network, given the data [23].

A Bayesian network (G, θ) is a representation of a joint probability distribution, where G is a graph and θ a probability distribution. G is a directed acyclic graph, such as the one on Fig. 2.2, where vertices correspond to network random variables.

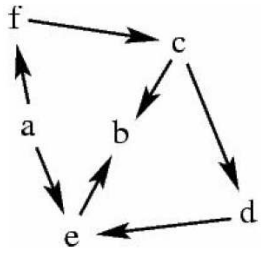


Figure 2.2: A directed acyclic graph.

These variables can be continuous or *discrete*. Directed edges correspond to dependencies between variables. θ describes a conditional distribution for each variable of the network, given its "parents" as defined by the relations in G. Together they capture the conditional independence relations between the variables.

2.3 Cellular Automata

Cellular automata were introduced in the 1940s by Stanislaw Ulam, who was studying crystal growth, and John von Neumann, working on self-reproduction in biology [24]. In the 1970s, John Conway's "Game of Life" [25], a two-state, two-dimensional cellular automaton, became widely known. In 1982 Stephen Wolfram published the first of a series of papers [26] systematically investigating a very basic class of cellular automata, which he terms *elementary cellular automata*. The unexpected complexity of the behavior of these simple rules led Wolfram to suspect that complexity in nature may be due to similar mechanisms.

A cellular automaton is a discrete model that consists of an infinite, regular grid of cells, each in one of a finite number of states. The grid can be in any finite number of dimensions. Time is also discrete, and the state of a cell at time t is a function of the states of a finite number of cells (called its neighborhood) at time $t-1$. These neighbors are a selection of cells relative to the specified cell, and do not change. Every cell has the same rule for updating, based on the values in this neighborhood. Each time the rules are applied to the whole grid a new generation is produced. See Fig. 2.3 for an example of a pattern constructed by a cellular automaton.

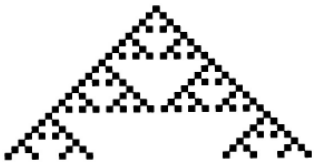


Figure 2.3: A pattern obtained with a simple cellular automaton: Pascal's triangle of binomial coefficients, reduced modulo 2. From [26].

Among other applications in systems biology, cellular automata are used to model reaction-diffusion systems [27] and biological cells [28].

2.4 Logical Models

In 1973 René Thomas introduced a detailed logical description of the mechanisms governing transcriptional regulation, including the effects of DNA domains such as promoters, initiators, terminators, and the concepts of genetic dominance and recessivity [30]. His logical framework was successfully applied to various gene regulatory networks playing a role in *Arabidopsis thaliana* [31] and *Drosophila melanogaster* [32].

Logical models use discontinuous variables and functions with limited number of values, often only two. To each variable one tries to attribute as few distinct values as there are qualitatively distinct levels. This emphasizes the essential qualitative features of the system at the expense of kinetic and mechanistic details. For example, if the biological role of a certain protein is to turn on a specific gene, one may consider that the protein is either present of absent, with the gene respectively on or off.

In its simplest form, a logical description associates a logical variable x with each element of the system. The variable takes value 1 when the real value exceeds a threshold value θ, and 0 otherwise. The state of the system can thus be described by a logical vector $(x, y, z, \dots)$, called a state vector. The evolution of the system is carried out by logical functions $X, Y, Z, \dots$ which are based on the interactions among the variables in such a way that a function takes value 1 for the conditions in which the corresponding variable tends to value 1 or remains at 1 [29].

In the classical logical description, time is introduced "synchronously"

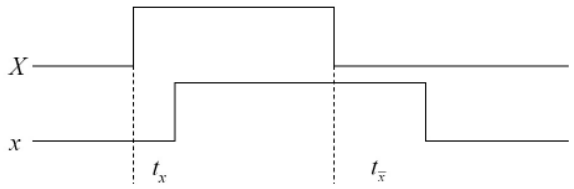


Figure 2.4: Temporal relationship between variable x and its image X.

by the next value $(x, y, z, \dots)_{t+1}$ for each value of the vector $(x, y, z, \dots)_t$. In kinetic logic, time is introduced through an "asynchronous" sequential description [30]. Let function X represent the state, on or off, of a gene, and variable x represent the presence or absence of the gene product. If the state of the system is such that the gene is switched on, *i.e.* X changes from 0 to 1, its product will be synthesized and, after a time delay t_x, it will reach its threshold concentration and x will change from 0 to 1. Similarly, when the gene is switched off (X changes from 1 to 0), the product will decay or be diluted out and, after another time delay $t_{\bar{x}}$, it will drop below its threshold concentration and x will switch from 1 to 0 (Fig. 2.4).

The formalism introduced by Thomas was later refined to include multilevel variables and used to study feedback loops, *e.g.* circular chains of interaction [29]. These loops can be classified into two categories based on the number of negative (inhibitory) interactions in the loop: if this number is even, the loop is positive, and if the number of negative interactions is odd, the loop is a negative feedback loop. Thomas found that a positive feedback loop is a necessary condition for the existence of multiple steady states, while a negative feedback loop with two or more elements is a necessary condition for stable limit cycles [29]. Biologically this means that cell differentiation is based on positive feedback loops, and homeostasis (stability to small perturbations) is based on negative feedback loops [33].

Chapter 3

Polynomial Models over Finite Fields

Constructing functions that fit exactly a given collection of points has long been a topic of interest in mathematics, statistics, and engineering, with cubic splines and power series being among the most common techniques for fitting points in $\mathbb{R}^n$. Since typically there are many functions that fit a set of points, the specific application determines which one provides the "best" fitting. For example, for many computer graphics uses where a smooth fitting curve is required, cubic splines are usually the optimal choice due to their ease of control. For some applications, however, it is necessary to describe *all* functions fitting a data set. Over an infinite field, this is possible only if some additional restrictions on the type of fitting functions are specified, such as whether they are polynomial or exponential, (piecewise) continuous, smooth, have certain derivatives, *etc.* This work concerns itself with polynomial functions vanishing on a set of data points. Given a finite set of points $V \in k^n$, where k is a field, we consider all polynomials in $k[x_1, \ldots, x_n]$ that vanish on the points in V. In Section 3.1.3 the construction of these polynomials will be discussed together with the fact that they form an *ideal* in $k[x_1, \ldots, x_n]$. Such ideals are fundamental for solving interpolation problems in numerical analysis [34], they are used in coding theory [35] and algebraic statistics [36].

3.1 Concepts from Commutative Algebra and Algebraic Geometry

This chapter introduces some basic terminology and facts from commutative algebra and algebraic geometry. The selection and organization of the material is determined by its need for the understanding of the results in Section 4 and is neither the complete nor traditional way of introducing the topics in it. For a comprehensive introductory treatment of the subject some excellent sources are [40], [41], and [42].

3.1.1 Gröbner Bases

The main motivation behind introducing Gröbner (Standard) Bases is that the remainder of long division of multivariate polynomials is not unquietly determined in general. It depends on the ordering of the polynomial terms (monomials). This is not an issue in univariate division where there is a unique monomial ordering and the division of polynomial f by polynomial g proceeds by dividing the highest power of the variable in g into the highest power in f. In other words, the one-variable monomials are ordered using degree ordering:

$$\cdots \succ x^{m+1} \succ x^m \succ \cdots \succ x^2 \succ x \succ 1.$$

With multivariate polynomials, however, there is more than one way of ordering their monomials and thus carry out long division. Consider, for example, the polynomials $f = x^2$ and $g = x^2 + xy^2 \in \mathbb{Q}[x, y]$. The remainder of dividing f by g is $-xy^2$ or x^2, depending on whether we choose the *initial monomial* of g to be $in(g) = x^2$ or xy^2.

Next we discuss the possible *monomial orderings* and how they impact polynomial division.

Monomial Ordering

A polynomial in $k[x_1, \ldots, x_n]$ is a linear combination of monomials of the form $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ over k, where α is the n-tuple exponent $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}^n_{\geq 0}$. For many purposes, it is necessary to be able to arrange the terms in a polynomial unambiguously in some order. This requires a *total* ordering on

the monomials, *i.e.* for every pair of monomials x^α and x^β, exactly one of the following must be true:

$$x^\alpha \prec x^\beta, \qquad x^\alpha = x^\beta, \qquad x^\alpha \succ x^\beta.$$

Taking into account the properties of the polynomial sum and product operations, the following definition emerges.

Definition 3.1.1 *A **monomial ordering** on $k[x_1, \ldots, x_n]$ is any relation $\succ$ on $\mathbb{Z}^n_{\geq 0}$ satisfying:*

1. *$\succ$ is a total ordering on $\mathbb{Z}^n_{\geq 0}$.*
2. *If $\alpha \succ \beta$ and $\gamma \in \mathbb{Z}^n_{\geq 0}$, then $\alpha + \gamma \succ \beta + \gamma$.*
3. *$\succ$ is a well-ordering on $\mathbb{Z}^n_{\geq 0}$, i.e. every nonempty subset of $\mathbb{Z}^n_{\geq 0}$ has a smallest element under $\succ$.*

One of the most popular monomial orderings is the *lexicographic ordering* which is analogous to the ordering of words in dictionaries and under which $x^2 \succ_{lex} xy^2$. Another one is the *graded lexicographic ordering* which orders by total degree first, then "breaks ties" using the lexicographic ordering. Under graded lexicographic ordering, $xy^2 \succ_{grlex} x^2$.

A monomial ordering can also be defined by a weight vector $\omega = (\omega_1, \ldots, \omega_n)$ in $\mathbb{Z}^n_{\geq 0}$. We require that ω have nonnegative coordinates in order for 1 to always be the smallest monomial. Fix a monomial ordering $\succ_\sigma$, such as $\succ_{lex}$. Then, for $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$, define $\alpha \succ_{\omega,\sigma} \beta$ if and only if

$$\omega \cdot \alpha \succ \omega \cdot \beta, \qquad \text{or} \qquad \omega \cdot \alpha = \omega \cdot \beta \qquad \text{and} \qquad \alpha \succ_\sigma \beta.$$

For example, weight vector $\omega = (3, 1)$ orders the monomials of polynomial $g = x^2 + xy^2$ in the same way as the lexicographic order, while $\omega' = (1, 1)$ performs the same monomial ordering as the graded lexicographic ordering.

Ideal Membership Problem

Another problem with multivariate polynomial division is that when dividing a given polynomial into more than one polynomials, the outcome may depend on the order in which the division is carried out. Let $f, h_1, \ldots, h_m \in k[x_1, \ldots, x_n]$ be polynomials in the variables $x_1, \ldots, x_n$. The so-called *ideal*

membership problem is to determine whether there are polynomials $h_1, \ldots, h_m \in k[x_1, \ldots, x_n]$ such that

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

To state this in the language of abstract algebra, we define

$$I = \langle f_1, \ldots, f_m \rangle = \left\{ \sum h_i f_i \mid h_1, \ldots, h_m \in k[x_1, \ldots, x_n] \right\}.$$

The polynomials in I form a so-called *ideal* in $k[x_1, \ldots, x_n]$ since they are closed under addition and multiplication by any polynomial in $k[x_1, \ldots, x_n]$. We ask whether f is an element of I. In general, even under a fixed monomial ordering, the order in which f is divided by the generating polynomials f_i affects the remainder $r_{\{f_i\}}(f)$. Therefore, $r_{\{f_i\}}(f) \neq 0$ does not imply $f \notin I$. Moreover, the generating set $\{f_1, \ldots, f_m\}$ of the ideal I is not unique but a special generating set $\mathcal{G} = \{g_1, \ldots, g_t\}$ can be selected so that the remainder of polynomial division of f by the polynomials in $\mathcal{G}$ performed in any order is zero if and only if f lies in I: $r_{\mathcal{G}}(f) = 0 \Leftrightarrow f \in I$. A generating set with this property is called a *Gröbner basis* and its precise definition will be given as Definition 3.1.3 after introducing the notion of initial ideals. Here we point out that Gröbner bases provide an algorithmic solution to the ideal membership problem and the Buchberger algorithm [43] is designed to compute a Gröbner basis for any ideal other than $\{0\}$ and a fixed monomial ordering.

3.1.2 Monomial Ideals

Gröbner bases are a key concept in computational algebra. Their theory reduces questions about systems of polynomial equations to the combinatorial study of *monomial ideals*.

Definition 3.1.2 *An ideal $I \subset k[x_1, \ldots, x_n]$ is a **monomial ideal** if there is a subset $A \subset \mathbb{Z}^n_{\geq 0}$ such that $I = \langle x^\alpha \mid \alpha \in A \rangle$, i.e. consists of all polynomials which are finite sums of the form $\sum_{\alpha \in A} h_\alpha x^\alpha$, where $h_\alpha \in k[x_1, \ldots, x_n]$.*

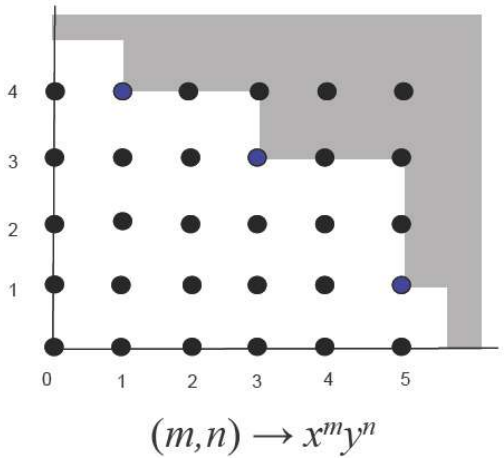


Figure 3.1: Staircase of monomial ideal $I = \langle x^2y^4, x^3y^3, x^5y \rangle$.

Staircases of Monomial Ideals

A monomial x^β is divisible by x^α exactly when $x^\beta = x^\alpha \cdot x^\gamma$ for some $\gamma \in \mathbb{Z}^n_{\geq 0}$, which is equivalent to $\beta = \alpha + \gamma$. The set

$$\left\{\alpha + \gamma \mid \gamma \in \mathbb{Z}^n_{\geq 0}\right\}$$

consists of the exponents of all monomials divisible by x^α. This allows us to visualize the monomials in a given monomial ideal. For example, the exponents of the monomials in $I = \langle x^2y^4, x^3y^3, x^5y \rangle \subset k[x, y]$ can be drawn as the union of the integer points in three translated copies of the first quadrant in the plane, as in Fig. 3.1. Such plots are called *staircases of monomial ideals*.

A special kind of monomial ideal is the *initial ideal* of an ideal $I \neq \{0\}$ for a fixed monomial ordering. It is the ideal generated by the set of initial monomials (under the specified ordering) of the polynomials of I:

$$in\,(I) = \langle in(f) \mid f \in I \rangle\,.$$

The monomials which do not lie in $in(I)$ are called *standard monomials*.

Definition 3.1.3 *Fix a monomial ordering. A finite subset $\mathcal{G}$ of an ideal I is a* ***Gröbner basis*** *if*

$$in(I) = \langle in(g) \mid g \in \mathcal{G} \rangle\,.$$

A Gröbner basis for an ideal may not be unique. If we also require that for any two distinct elements $g, g' \in \mathcal{G}$, no term of g' is divisible by $in(g)$, such a Gröbner basis is called *reduced* and is unique for an ideal and a term ordering, provided the coefficient of $in(g)$ in g is 1 for each $g \in \mathcal{G}$.

The Gröbner Fan of an Ideal

Over a quotient polynomial ring
$k[x_1, \ldots, x_n]/\langle x_1^p - x_1, \ldots, x_n^p - x_n \rangle$ with $|k| = p$, there are p^n monomials and thus, $p^n!$ possible monomial orderings. A fixed ideal, however, has only a finite number of different reduced Gröbner bases. Informally, the reason is that most of the monomial orderings only differ in high degree and the Buchberger algorithm for Gröbner basis computation does not "see" the difference among them. For many applications the appropriate choice of a monomial ordering for computing the Gröbner basis of an ideal is unclear, while its impact is essential. Consequently, one may want to consider *all* reduced Gröbner bases of a given ideal I. This is equivalent to studying the different initial ideals because the monomial initial ideals of I are in bijection with the marked reduced Gröbner bases of I. A *marked* Gröbner basis is a set of polynomials which is a Gröbner basis with respect to some monomial ordering with the initial term of each polynomial being distinguished.

A combinatorial structure that contains information about the initial ideals of an ideal is the *Gröbner fan* of an ideal. It is a polyhedral complex of cones, each corresponding to an initial ideal, such as the Gröbner fan in Fig. 3.2.

For details on the Gröbner fan and its computation see, for example, [44]. For the purposes of the current work it is enough to know that there are algorithms based on the Gröbner fan that enumerate all reduced Gröbner bases of a polynomial ideal. An excellent implementation of such an algorithm is the software package `Gfan` [45] that we have used extensively in our work.

3.1.3 Ideals of Points

Given a set of points, it is often necessary to find all the polynomials that vanish on it. Such a set of polynomials forms an *ideal of points* defined as follows.

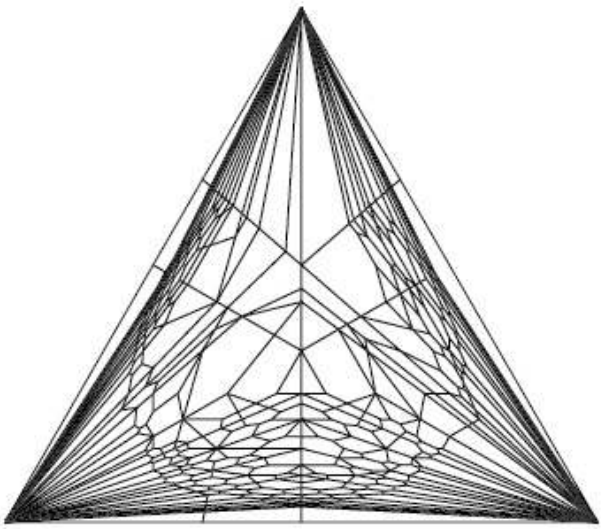


Figure 3.2: A Gröbner fan of an ideal.

Definition 3.1.4 *Let* $V = \{p_1, \ldots, p_m\}$, *where* $p_i = (a_{i1}, \ldots, a_{in}) \in k^n$. *Then we set*

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

It can be shown that $\mathcal{I}(V)$ *is an ideal of* $k[x_1, \ldots, x_n]$. *It is called the* ***ideal of points*** *in* V.

To see that $\mathcal{I}(V)$ is indeed an ideal and to obtain an algorithm for computing such ideals, first notice that the ideal $\mathcal{P}_i$ of all polynomials that vanish on a single point $p_i = (a_{i1}, \ldots, a_{in}) \in k^n$ contains the ideal $I = \langle x_1 - a_{i1}, \ldots, x_n - a_{in} \rangle$. But this ideal is *maximal* with respect to inclusion (see [40], Proposition 9 on p. 198 for a proof) in the sense that any ideal J containing I is such that either $J = I$ or $J = k[x_1, \ldots, x_n]$ and therefore

$$\mathcal{P}_i = \langle x_1 - a_{i1}, \ldots, x_n - a_{in} \rangle .$$

The intersection of all ideals $\mathcal{P}_i$ is an ideal itself and contains exactly the polynomials that vanish on the points in V:

$$\mathcal{I}(V) = \bigcap_{i=1}^{m} \mathcal{P}_i.$$

Intersections of ideals can be computed algorithmically (see [40], p. 185).

For the results presented in the Section 4, it is essential to notice that the number of standard monomials in $in(I)$ is equal to the number of points in V. This result is presented in the following proposition.

Proposition 3.1.5 *Let $I = \mathcal{I}(V)$ be the ideal of the points in $V = \{(a_{11}, \ldots, a_{1n}), \ldots, (a_{v1}, \ldots, a_{vn})\} \subset k^n$. The number of standard monomials in any $in(I)$ is equal to the number of points in V.*

Proof: Let I be an ideal of the polynomial ring $k[x_1, \ldots, x_n]$ with $I = \bigcap_{i=1}^{r} I_i$, where I_i are pairwise *comaximal* (meaning $I_i + I_j = k[x_1, \ldots, x_n]$ for all $i \neq j$). By the Chinese Remainder Theorem,

$$\frac{k[x_1, \ldots, x_n]}{I} \cong \prod \frac{k[x_1, \ldots, x_n]}{I_i}.$$

$k[x_1, \ldots, x_n]/I$ is isomorphic as a k-vector space to $\mathrm{Span}(x^\alpha \mid x^\alpha \notin in(I))$ (see [40], p. 229, Proposition 4). Hence so is $k[x_1, \ldots, x_n]/in(I)$ and thus $k[x_1, \ldots, x_n]/I \cong k[x_1, \ldots, x_n]/in(I)$. If I is an ideal of points, then $I_i = \langle x_1 - a_{i1}, \ldots, x_n - a_{in} \rangle$ are comaximal (*without* assuming that k is algebraically closed), and therefore

$$\frac{k[x_1, \ldots, x_n]}{in(I)} \cong \prod \frac{k[x_1, \ldots, x_n]}{\langle x_1 - a_{i1}, \ldots, x_n - a_{in} \rangle}. \tag{3.1}$$

The dimension of the left-hand side of (3.1) is the number of standard monomials of I and the dimension of the right-hand side is the number of points in V. Therefore, the two are equal. □

3.2 Polynomial Dynamical Systems

In this section we explore time-discrete multistate dynamical systems. These systems constitute the modeling framework of the reverse engineering method developed for gene regulatory networks from DNA microarray data in [47], which will be outlined next in Section 3.3.

Definition 3.2.1 *Let X be a finite set. A **finite dynamical system** of dimension n is a function $F = (f_1, \ldots, f_n) : X^n \to X^n$ with $f_i : X^n \to X$.*

By requiring that the cardinality of the set X be a power of a prime number, one can impose on X the structure of a *finite field*. This structure determines the only type of functions f_i that need to be considered. The following theorem ([48]) characterizes functions over finite fields.

Theorem 3.2.2 *Let k be a finite field. Then every function $f : k^n \to k$ is a polynomial of degree at most n.*

The reverse engineering method of [47] is interested mostly in the case when k is a finite field of cardinality p, *i.e.* $k = \mathbb{F}_p$, and thus the polynomials f_i are in the quotient ring $R = k[x_1, \ldots, x_n]/\langle x_1^p - x_1, \ldots, x_n^p - x_n \rangle$. As a result, they are polynomials in n variables with coefficients in k and the degree of each variable is at most equal to $p - 1$.

Definition 3.2.3 *If the set X for a finite dynamical system is a finite field, then F is called a* ***polynomial dynamical system*** *and often X is denoted by k to distinguish it as a finite field.*

3.3 An Algebraic Approach to Reverse Engineering

In their paper, [47], Laubenbacher and Stigler propose a new modeling approach that describes a regulatory network as a polynomial dynamical system. The method is further developed in [49]. A *gene regulatory network* is a collection of genes that interact with each other, thereby governing the rates at which the genes in the network are transcribed. The objective of the method is to discover regulatory relationships among the nodes in the network from *discrete* data. The next paragraph contains an outline of this process of reverse-engineering for building polynomial dynamical systems given time courses of discrete data. The algorithm constructs the set of all polynomial dynamical systems for the data and then uses a minimality criterion to select one system from the set. A unique feature of this method is the ability to construct all polynomial dynamical systems that satisfy the given time courses. This is not done via enumeration, but rather it is accomplished by way of Gröbner bases, discussed in Section 3.1.1.

Let the finite field k be the set of possible states of the nodes of the network and let $f : k^n \to k^n$ be a polynomial dynamical system of dimension

n. Then f can be described in terms of its coordinate functions $f_i : k^n \to k$, for $i = 1, \ldots, n$. That is, if $\mathbf{x} = (x_1, \ldots, x_n) \in k^n$ is a state, then $f(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}))$. The reverse-engineering problem of interest is, given one or more time courses of state transitions generated by a biological system with n varying quantities, choose a polynomial dynamical system $f : k^n \to k^n$ which fits the data and "best describes" the biological system. It is assumed that a set of state transitions of the network is given in the form of one or more time courses of network states

$$\begin{array}{rclcrcl} \mathbf{s}'_1 & = & (s'_{11}, \ldots, s'_{n1}), \ldots, & \mathbf{s}'_m & = & (s'_{1m}, \ldots, s'_{nm}) \\ \mathbf{s}''_1 & = & (s''_{11}, \ldots, s''_{n1}), \ldots, & \mathbf{s}''_r & = & (s''_{1r}, \ldots, s''_{nr}) \end{array}$$

$$\vdots$$

satisfying the property that, if the unknown transition function of the network is f, then

$$\begin{array}{l} f(\mathbf{s}'_i) = \mathbf{s}'_{i+1}, \quad i = 1, \ldots, m-1 \\ f(\mathbf{s}''_j) = \mathbf{s}''_{j+1}, \quad j = 1, \ldots, r-1 \end{array}$$

$$\vdots$$

Unless all state transitions of the system are specified, there will be more than one network that fits the given data set. Since this much information is hardly ever available in practice, any reverse-engineering method has to choose from a large set of possible network models. First, the space of all networks that are consistent with the given time course data is computed. Then a particular network $f = (f_1, \ldots f_n)$ is chosen that satisfies the following property:

For each i, f_i is *minimal* in the sense that there is no non-zero polynomial $g \in k[x_1 \ldots x_n]$ such that $f = h + g$ and g is identically equal to zero on the given time points.

The above criterion for model selection is analogous to the problem of excluding the terms of f_i that vanish on the data. The advantage of the polynomial modeling framework over a finite field is that there is a well-developed algorithmic theory that provides mathematical tools for the solution of this problem.

3.3.1 Algorithm for Model Selection (for one time course)

Input: A time course of network states $\mathbf{s}_1, \ldots \mathbf{s}_m \in k^n$ and expression levels $a_1, \ldots, a_m \in k$.

Output: All polynomial functions $f \in k[x_1, \ldots, x_n]$ such that $f(s_j) = a_j$ for all $j = 1, \ldots, m$ such that f does not contain component polynomials that vanish on the time course .

Step 1. Compute a particular polynomial f_0 that fits the data.
There are several methods to do this, Lagrange interpolation being one of them. [47] uses the following formula, based on the Chinese Remainder Theorem:

$$f_0(\mathbf{x}) = \sum_{j=1}^{m} a_j r_j(\mathbf{x}),$$

with the polynomials r_j defined as follows. Let $1 \leq i \neq j < m$. If $\mathbf{s}_i \neq \mathbf{s}_j$, then find the first coordinate l in which they differ. Define

$$b_{ij}(\mathbf{x}) = (s_{jl} - s_{il})^{p-2}(x_l - s_{il})$$

for every $i \neq j$. For all $i \neq j$, define

$$r_j(\mathbf{x}) = \prod_{i=1}^{m-1} b_{ij}(\mathbf{x}).$$

It can be easily verified that this polynomial does indeed interpolate $\mathbf{s}_1, \ldots \mathbf{s}_m$.

Step 2. Compute the ideal I of all functions that vanish on the data.
Notice that if two polynomials $f, g \in k[x_1, \ldots, x_n]$ such that $f(\mathbf{s}_j) = a_j = g(\mathbf{s}_j)$, then $(f - g)(\mathbf{s}_j)$ for all j. Therefore, in order to find all functions that fit the data, we need to find all functions that vanish on the given time points. As discussed in Section 3.1.3, those functions form an ideal of points and can be computed algorithmically.

Step 3. Reduce the polynomial f_0 found in Step 1 modulo the ideal I. That is, write f_0 as $f_0 = f + g$ with $g \in I$ and f being minimal in the sense that it cannot be further decomposed into $f = f' + h$ with $h \in I$. In

other words, g represents the part of f_0 that lies in I, and that therefore is identically equal to 0 on the given time course. All possible functions that interpolate the time course are obtained in the form $f + g$, where g runs through all elements of I.

In the next two sections we present two problems that the reverse-engineering method described above faces.

3.3.2 Reverse Engineering Dependency on Monomial Ordering

The first problem originates from Step 2 of the reverse-engineering algorithm in Section 3.3.1: finding all polynomials that vanish on a set of points. This is equivalent to computing the ideal of these points and as we saw in Section 3.1.3, computation of an ideal of points boils down to intersection of ideals. There is a well-known consequence of the Buchberger Algorithm, originally presented in [37], for their computation. More efficient ways are the BM-algorithm in [38] and the modular BM-algorithm [39]. For an ideal $I \subset k[x_1, \ldots, x_n]$, the output of these algorithms is a finite set of polynomials $g_1, \ldots, g_s \in I$ that generate I:

$$I = \langle g_1, \ldots, g_s \rangle = \sum_{i=1}^{s} h_i g_i,$$

where $h_i \in k[x_1, \ldots, x_n]$. The existence of a finite generating set for any polynomial ideal is guaranteed by the Hilbert Basis Theorem. The generating set, however, is not unique. The type of generating sets called Gröbner bases, discussed in Section 3.1.1, possesses some nice properties but their computation depends on the choice of **monomial ordering**. For different monomial orderings the resulting Gröbner bases may vary greatly. In the same way, the computation of a Gröbner basis for an ideal of points also depends on the choice of monomial ordering. In Section 4 we present results on the relationship between the points in a set $V \subset k^n$ and the number of reduced Gröbner bases/monomial ideals for $\mathcal{I}(V)$.

3.3.3 Reverse Engineering Dependency on Data Discretization

The second problem lies in the data preprocessing stage. Since gene expression data are real numbers (represented by computer floating point numbers), the first step in any reverse-engineering algorithm using discrete models must be to **discretize** these numbers into a finite (typically small) set of possible states. Obviously, the way in which one discretizes the data plays an important role in what model one obtains. The first important choice is the number of discrete states allowed, the choice of p in the above algorithm. A major challenge in discretizing biological data, such as microarray experiments, is the typically small samples of data. Many methods for discretization are not applicable due to the insufficient amount of data. In Section 5 we present a first attempt to develop a discretization tool that takes into consideration the issues and limitations that are inherent in short time courses. Our focus is on the two characteristics that any discretization method should posses in order to be used for dynamic modeling: preservation of dynamics and information content and noise inhibition.

Chapter 4

Gröbner Bases for Ideals of Points

4.1 Motivation

The Buchberger algorithm computes the reduced Gröbner basis of a polynomial ideal with respect to a prescribed monomial ordering. The appropriate choice of a monomial ordering may be dictated by the application, as it is the case with the reverse lexicographic and elimination orderings which play an important role in elimination theory [46]. In other cases, however, there is insufficient information to make this choice.

Example 4.1.1 *Let* $f = x^2 \in \mathbb{F}_3[x,y]/\langle x^3 - x, y^3 - y\rangle$. *Consider computing the minimal form of* f *with respect to the set of points* $V = \{(2,0),(0,1)\} \subset \mathbb{F}_3^2$.
The ***minimal form*** *of* f *is such that there is no non-zero polynomial* $g \in \mathcal{I}(V)$ *with* $f = h + g$. *This means that it should not be divisible by any polynomial in* $I = \mathcal{I}(V) = \langle x-2, y\rangle \cap \langle x, y-1\rangle$. *To assure that, one takes the minimal form of* f *to be the unique remainder of* f *obtained after dividing* f *in some order by the generators of* $\mathcal{G}(I)$, *the reduced Gröbner basis of* I. *The result* $\overline{f}^{\mathcal{G}(I)}$ *is called the* ***normal form*** *of* f *with respect to* $\mathcal{G}(I)$ *and it may depend on the term ordering under which* $\mathcal{G}(I)$ *was computed.*
The ideal of points I *has two distinct reduced Gröbner bases:*

$$\mathcal{G}_1(I) = \left\{x - y + 1, y^2 - y\right\} \quad \textit{and} \quad \mathcal{G}_2(I) = \left\{x^2 + x, y - x + 2\right\}$$

which give two different normal forms for f:

$$\overline{f}^{\mathcal{G}_1(I)} = -y + 1 \quad \textit{and} \quad \overline{f}^{\mathcal{G}_2(I)} = -x.$$

In the above example the two minimal forms of the polynomial look quite different and applications, such as the reverse engineering in [47] that uses polynomial dynamical systems for modeling, strongly depend on selecting the minimal form of a polynomial. Typically, it is hard to determine which monomial ordering to use. Instead of making an arbitrary choice, we propose a method of overcoming this computational artifact. It finds all possible minimal sets of additional points W to be added to V so that $\mathcal{I}(V \cup W)$ has a unique Gröbner basis (*i.e.*, each Gröbner basis is universal).

4.2 Existence of a Solution

Given a set of points in k^n, the number of reduced Gröbner bases that the ideal of these points has is not obvious in general. For the two trivial cases, when only a single point is given, $V_1 = (a_1, \ldots, a_n) \in k^n$, and when we have all the points, $V_{k^n} = k^n$, it is clear that the ideals of points have unique Gröbner bases: $\mathcal{G}(V_1) = \{x_1 - a_1, \ldots, x_n - a_n\}$ and $\mathcal{G}(V_{k^n}) = \{x_1^p - x_1, \ldots, x_n^p - x_n\}$. The number of Gröbner bases in the intermediate cases is not so apparent. A set of nine points in $\mathbb{F}_3^3$, for example, can yield a single Gröbner basis or as many as twelve Gröbner bases, depending on which points exactly are taken. From the second trivial case, when all points in k^n are given, it is clear that it is always possible to add enough points to the given ones in V and obtain a single Gröbner basis: simply add the rest of the points, $k^n \setminus V$. In principle, one can also remove points from the given set in order to change the number of Gröbner bases and again, removing many enough would eventually produce a single Gröbner basis. In the current work, however, we do not consider this possibility due to our motivating application: each point in the given data set carries valuable information, which may cost hundreds of dollars, as in the case of microarray experiments. Not utilizing it would mean deliberately ignoring useful information. For this reason, the rest of the work is concerned only with the question

Given a set of points V such that $\mathcal{I}(V)$ has more than one Gröbner basis, what minimal subsets $W \subset k^n \setminus V$ can be added to V so that $\mathcal{I}(V \cup W)$ has a unique Gröbner basis?

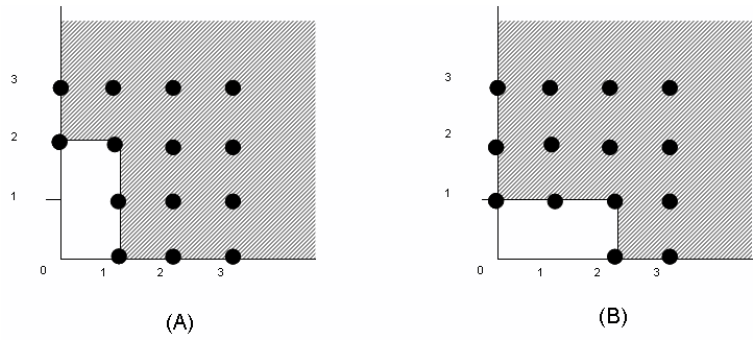


Figure 4.1: Monomial staircases of the initial ideals of I, $(m, n) \to x^m y^n$.
(A) $in_1(I) = \langle x, y^2 \rangle$ (B) $in_2(I) = \langle x^2, y \rangle$.

4.3 Cancellation of Monomial Ordering Effect on Gröbner Basis Computation

In this section we present an approach to eliminating the effect of different monomial orderings on the computation of the Gröbner basis of an ideal of points through adding points to the set of given points. The approach is based on excluding the polynomials whose leading monomials under some ordering cause the existence of multiple initial ideals (and consequently – multiple Gröbner bases) by adding points at which they do not vanish.

4.3.1 Elimination of Initial Terms through One Point Addition

We begin by showing how to find, if it exists, a single point which, when added to the set of given points, yields an ideal of points that has a unique Gröbner basis. The method can be generalized to adding more than one point with the same purpose as will be demonstrated in the subsequent section. We use the ideal of points from Example 4.1.1 for illustration and summarize the result in Lemma 4.3.1.

Consider again the ideal of points I from Example 4.1.1. The initial ideals corresponding to the two reduced Gröbner bases are given on Fig. 4.1.

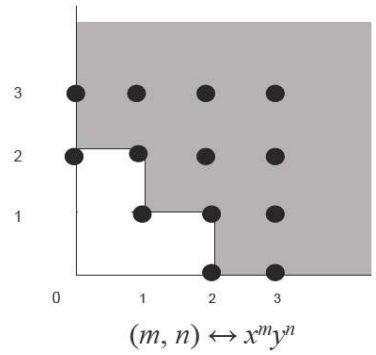


Figure 4.2: Monomial staircase of $in_1(I) \cap in_2(I) = \langle x^2, xy, y^2 \rangle$.

$$in_1(I) = \left\langle x, y^2 \right\rangle \quad \text{and} \quad in_2(I) = \left\langle x^2, y \right\rangle .$$

It is straightforward to see that if $W \subset \mathbb{F}_3^2$ is such that $J = \mathcal{I}(V \cup W)$ has a unique Gröbner basis and thus $in(J)$ is the same for any monomial ordering, then

$$in(J) \subseteq in_1(I) \cap in_2(I) = \left\langle x^2, xy, y^2 \right\rangle .$$

The monomial staircase of $in_1(I) \cap in_2(I)$ is given on Fig. 4.2. Notice that $in_1(I) \cap in_2(I)$ contains exactly one standard monomial more than $in_1(I)$ and $in_2(I)$: x and y, respectively. Therefore, in order for $x, y \notin in(J)$, the new ideal J should not contain any polynomials f with $in(f) = x$ or y with respect to *any* monomial ordering. To eliminate such polynomials, it is necessary to add point(s) to V at which these polynomials do not vanish. Recalling that the number of standard monomials in the initial ideal of an ideal of points is equal to the number of points over which the ideal is constructed, it follows that at least one point should be added to V in order to eliminate monomials x and y from $in(J)$ (equivalently, make x and y standard monomials of J) and thus obtain $in(J) = \langle x^2, xy, y^2 \rangle$. In general, it is possible that V is such that adding a single point, $W = \{(a_1, a_2)\}$, would not yield an ideal $\mathcal{I}(V \cup W)$ with a unique Gröbner basis for any $a_1, a_2 \in \mathbb{F}_3$. In such a case, while monomials x and y should still be eliminated from the initial ideal of J, other monomials would also have to be selected from $in(I)$.

Let us assume that there is a point $(a_1, a_2) \in \mathbb{F}_3^2$ such that for each $f \in J$ with $in(f) = x$ or y with respect to some monomial ordering, $f(a_1, a_2) \neq 0$,

and attempt to find it. Any function that can have x as an initial term under some monomial ordering is of the form

$$f_1 = k_{11}x + k_{12}y + k_{13}y^2 + k_{14},$$

where the coefficients $k_{1j} \in \mathbb{F}_3$. Such a function can have as non-initial terms only terms that are not divisible by x. The reason is that otherwise no monomial ordering would produce $in(f_1) = x$ since that would imply that 1 is not the smallest term. Another condition on the functions f_1 is that they are in the ideal of points $I = \mathcal{I}(V)$ and so must vanish on the points in V. Similarly, a function with y as an initial term under some monomial ordering should be of the form

$$f_2 = k_{21}y + k_{22}x + k_{23}x^2 + k_{24}$$

and also vanish on the points in V. Without loss of generality, assume that x and y's coefficients are 1 and solving $f_1(2,0) = 0 = f_1(0,1)$ and $f_2(2,0) = 0 = f_2(0,1)$ for the coefficients k_{ij}, gives all six polynomials that are in J and have x or y as an initial term under some monomial ordering:

$$\begin{aligned} f_{11} &= x + 2y + 1, \quad f_{12} = x + y + y^2 + 1, \\ f_{13} &= x + 2y^2 + 1, \quad f_{21} = y + x^2 + 2, \\ f_{22} &= y + x + 2x^2 + 2, \quad f_{23} = y + 2x + 2. \end{aligned} \tag{4.1}$$

Among the points in $\mathbb{F}_3^2 \setminus V$, only $(a_1, a_2) = (0,0)$ and $(2,1)$ are such that none of f_{ij} vanishes on (a_1, a_2). By inspection, one can verify that indeed these two points are the only ones that, added separately to V, generate an ideal of points with a unique Gröbner basis. Notice that depending on which point is added to V, a different Gröbner basis is produced but in either case the initial ideal is $\langle x^2, xy, y^2 \rangle$.

The above reasoning can be generalized for an ideal of points over any finite field, as in the following lemma. In the example, $m = n = 2$, $V = \{(2,0),(0,1)\}$, $M = \{x, y\}$, $\mathcal{F}$ consists of the polynomials in (4.1), and $(a_1, a_2) = (0,0)$ or $(2,1)$.

Lemma 4.3.1 *Let k be a finite field and $V \subset k^n$ be a set of points for which $I = \mathcal{I}(V)$ has initial ideals $in_1(I), \ldots, in_m(I)$ in the polynomial ring $R = k[x_1, \ldots, x_n] / \langle x_1^p - x_1, \ldots, x_n^p - x_n \rangle$. Suppose that the number of standard monomials in each $in_j(I)$ is one less than this number in $\bigcap_{i=1}^m in_i(I)$.*

Let M be the set of monomials x^{α_i} that are standard in $\bigcap_{i=1}^{m} in_i(I)$ but not in $in_j(I)$ for some j. Finally, let

$$\mathcal{F} = \{f \in R \mid f(p) = 0 \ \forall \ p \in V \text{ and } in(f) = x^{\alpha_i} \in M \text{ for some monomial ordering}\}.$$

Then $\mathcal{I}(V \cup \{(a_1, \ldots, a_n)\})$, $a_i \in k$, has a unique Gröbner basis if and only if for any $f \in \mathcal{F}$, $f(a_1, \ldots, a_n) \neq 0$.

4.3.2 Elimination of Initial Terms through Multiple Point Addition

Lemma 4.3.1 is useful for illustrating the principle of eliminating initial monomials through adding points but for larger examples it would be often necessary to include more than one extra point to eliminate the effect of monomial ordering and obtain a single Gröbner basis. The following is a generalization for these cases.

Theorem 4.3.2 *Let*

$$\mathcal{F} = \{f \in R \mid f(p) = 0 \ \forall \ p \in V \text{ and } in(f) = x^{\alpha_i} \in M \text{ for some monomial ordering}\}.$$

Over R, $\mathcal{F}$ contains finitely many polynomials. For a set of points $W \subset k^n \backslash V$ to be such that $\mathcal{I}(V \cup W)$ has a unique Gröbner basis, W must satisfy

1. *$\mathcal{F} \cap \mathcal{I}(W) = \emptyset$;*
2. *the number of points in W,*
   $|W|$ = (number of standard monomials in $\bigcap_{i=1}^{m} in_i(I)$) – (number of standard monomials in $in(I)$).

Proof: The first condition guarantees that for each $f \in \mathcal{F}$, there is at least one point p in W such that $f(p) \neq 0$ and so f will not be in $\mathcal{I}(V \cup W)$. The latter condition follows from Proposition 3.1.5. □

Although these two conditions are both necessary and sufficient, generating W by explicitly finding the polynomials in $\mathcal{F}$ is impractical since the number of polynomials can easily be in the hundreds for larger examples. Another problem with applying the above approach directly is in actually finding the points in W that satisfy the two conditions, especially since we want to be able to find all such minimal sets W. For small fields, such as

the one of Example 4.1.1, checking all possibilities is an option but for larger fields this would be computationally inefficient. We present an algorithm for finding the sets of points in W. The algorithm is based on the fact that Gröbner bases solve the ideal membership problem and can be used to determine whether a polynomial is an element of an ideal. The Gröbner bases computation can be carried out with respect to whichever monomial ordering is efficient for the particular algebra software package of choice.

Method for finding W. Recall that each polynomial in $\mathcal{F}$ has as a leading term under some monomial ordering one of the terms x^{α_j} to be eliminated from $in(\mathcal{I}(V \cup W))$. For a fixed term x^{α_j}, what are the polynomials f that have $in(f) = x^{\alpha_j}$ under some monomial ordering? Clearly, they should be of the form

$$x^{\alpha_j} + \sum_{\beta_j} c_{\beta_j} x^{\beta_j}, \tag{4.2}$$

where $c_{\beta_j} \in k$ and $x^{\alpha_j} \nmid x^{\beta_j}$. Additionally, in order for c_{β_j} to be the leading term under some ordering, some combinations of other terms should not be permitted. For example, over $k[x, y]$, xy cannot be the leading term of polynomial $f = xy + x^2 + y^2$ because under any monomial ordering, $in(f)$ is either x^2 or y^2. Therefore, we should require the following condition on the terms x^{β_j} from (4.2). Let $\alpha_j = (\alpha_{j1}, \ldots, \alpha_{jn})$ and $\beta_j^l = (\beta_{j1}^l, \ldots, \beta_{jn}^l)$. Consider all sets of n terms $\{\beta_j^1, \ldots, \beta_j^n\}$ such that $|\beta_j^l| = |\alpha_j|$ for each $l = 1, \ldots, n$ and $\alpha_{jl} < \beta_{jl}^l$. Require that if $c_{\beta_j^l} \neq 0$ for some l, then $c_{\beta_j^m} = 0$ for some $m \neq l$. This, together with (4.2), generates the set of all polynomials with leading term x^{α_j} under some monomial ordering. For each term α_j, denote this set by $\mathcal{F}_{\alpha_j}$.

We now proceed to finding the points in

$$W = \{(a_{11}, \ldots, a_{1n}), \ldots, (a_{s1}, \ldots, a_{sn})\}$$

where $s = |V \cup W|$. Fix a convenient monomial ordering. Let $\mathcal{G}$ be the Gröbner basis of $I = \mathcal{I}(V)$. Let $I_i = \mathcal{I}(\{(a_{i1}, \ldots, a_{in})\})$, the ideal of polynomials that vanish on point $(a_{i1}, \ldots, a_{in}) \in W$, with $\mathcal{G}_i$ being the Gröbner basis of I_i. Hence $\mathcal{G}_i = \{x_1 - a_{i1}, \ldots, x_n - a_{in}\}$. If for all $f \in \mathcal{F}_{\alpha_j} \subset \mathcal{F}$, $\overline{f}^{\mathcal{G}_i} \neq 0$ for some i and j, then $f \notin I_i$ for some i and so $f \notin \bigcap_{i=1}^{s} I_i = \mathcal{I}(W)$ which means that $\mathcal{F} \cap \mathcal{I}(W) = \emptyset$. In the calculation of $\overline{\mathcal{F}_{\alpha_j}}^{\mathcal{G}_i}$ below, $a_{i1}, \ldots, a_{in} \in k$

are kept symbolic and the remainder $\overline{\mathcal{F}_{\alpha_j}}^{\mathcal{G}_i}$ is obtained as a function of them and the c_{β_j}'s by consecutively dividing $\mathcal{F}_{\alpha_j}$ by $x_l - a_{kl}$. Therefore, for each $\mathcal{F}_{\alpha_j} \subset \mathcal{F}$, one should consider the system of equations

$$\begin{cases} \overline{\mathcal{F}_{\alpha_j}(x_1, \ldots, x_n)}^{\mathcal{G}} & = 0 \\ \overline{\mathcal{F}_{\alpha_j}(a_{11}, \ldots, a_{1n})}^{\mathcal{G}_1} & = 0 \\ \vdots & \\ \overline{\mathcal{F}_{\alpha_j}(a_{s1}, \ldots, a_{tn})}^{\mathcal{G}_t} & = 0 \end{cases} \tag{4.3}$$

and find all subsets of points $S_{\alpha_j} = \left\{S_{\alpha_j i}\right\}$ with

$$S_{\alpha_j i} = \{(a_{11}, \ldots, a_{1n}), \ldots, (a_{s1}, \ldots a_{sn})\} \subset k^n$$

that make (4.3) inconsistent, selecting t as large as necessary. For each α_j, select i such that $\left|\bigcap_{\alpha_j} S_{\alpha_j i}\right| = s$. Take $W = \bigcap_{\alpha_j} S_{\alpha_j i}$.

Outline of the algorithm for finding W:

Input: $V \subset k^n$, $I = \mathcal{I}(V)$ with multiple distinct Gröbner bases, $\mathcal{G}$ a Gröbner basis of I with respect to some convenient monomial ordering, M and $\mathcal{F}$ as defined in Lemma 4.3.1.

Output: $W \subset k^n \setminus V$ such that $\mathcal{I}(V \cup W)$ has a unique Gröbner basis.

1. Determine $s = |W|$. Denote $W = \{(a_{11}, \ldots, a_{1n}), \ldots, (a_{s1}, \ldots, a_{sn})\}$.
2. For each $i = 1, \ldots, s$, generate

   $$I_i = \mathcal{I}(\{(a_{11}, \ldots, a_{1n})\})$$

   and its Gröbner basis

   $$\mathcal{G}_i = \{x_1 - a_{i1}, \ldots x_n - a_{in}\}.$$

3. Let $\mathcal{F}_{a_j} = \{f \in \mathcal{F} \mid in(f) = x^{a_j} \in M$.
4. For each $\mathcal{F}_{a_j}$, find the set of points S_{a_j} that makes system (4.3) inconsistent.
5. Choose $W = \cap S_{a_j}$.

4.4 Nonexistence of Solution for a Fixed $|W|$

As it was mentioned in Section 4.3.1, the second condition of Theorem 4.3.2 is not always possible to satisfy. In other words, given a set of points $V \subset k^n$ with an ideal $I = \mathcal{I}(V)$ which has distinct initial ideals $in_1(I), \ldots, in_m(I)$ with respect to some monomial orderings, it is not always true that there exists a subset of points $W \subset k^n \setminus V$ with

$$\begin{aligned} |W| = &(\text{number of standard monomials in } \textstyle\bigcap_{i=1}^{m} in_i(I)) \\ &- (\text{number of standard monomials in } in(I)) \end{aligned}$$

such that $in(\mathcal{I}(V \cup W)) = \cap_{j=1}^{m} in_j(I)$. In the light of Proposition 3.1.5 this may seem surprising but the following simple counterexample is a good illustration of the fact that the right-hand side of the above equation is only a lower bound.

Example 4.4.1 *Let $R = \mathbb{F}_3[x,y]/\langle x^3 - x, y^3 - y\rangle$ and $V = \{(0,0),(1,1),(2,2),(0,1)\}$. $I = \mathcal{I}(V)$ has two Gröbner bases*

$$\begin{aligned} \mathcal{G}_1(I) &= \{x^2 - y^2 - x + y, xy - y^2 - x + y, y^3 - y\}, \\ \mathcal{G}_2(I) &= \{x^3 - x, xy - x^2, y^2 - x^2 + x - y\} \end{aligned}$$

and so the initial ideals of I are

$$in_1(I) = \left\langle x^2, xy, y^3 \right\rangle \quad \text{and} \quad in_2(I) = \left\langle x^3, xy, y^2 \right\rangle.$$

Let $J = in_1(I) \cap in_2(I) = \langle x^3, xy, y^3\rangle$. Since both $in_1(I)$ and $in_2(I)$ have four standard monomials while J has five, one may expect that there is a single point $(a,b) \in k^2 \setminus V$ such that

$$in(\mathcal{I}(V \cup \{(a,b)\})) = J.$$

However, by inspection one can verify that none of the five points in $k^2 \setminus V$ satisfies the above equality. In fact, $\mathcal{I}(V \cup \{(a,b)\})$ has two Gröbner bases for any $a, b \in k$.

To solve this problem, one cannot simply add more points to W. From Theorem 3.1.5 we know that adding a point results in increasing the number of standard monomials by one and this may potentially increase the number

of distinct initial ideals for the ideal and consequently – the number of its Gröbner bases. To avoid that, we choose another monomial x^{γ_1} that is in all $in_i(I)$ and "on the verge" of the monomial staircase and add to the polynomials to be eliminated the ones that can have x^{γ_1} as a leading term. If still no solution exists, choose another x^{γ_2} and so on. As demonstrated in Section 4.2, there is an upper bound on the number of terms x^{γ_j} that need to be added to obtain a single Gröbner basis for the ideal of points.

4.5 Conclusion

The proposed method provides an algorithm for determining the minimal sets of data points to be added to a given set of points in order for the ideal of points of the new set to have a unique Gröbner basis. Mathematically, the theoretical grounds of the method establish a relationship between sets of points in k^n and number of reduced Gröbner bases/initial ideals of the ideals of points over $k[x_1, \ldots, x_n]/\langle x_1^p - x_1, \ldots, x_n^p - x_n \rangle$, which can be thought of as a measure of their dependence on the choice of monomial ordering.

In applications, this means being able to tell an experimentalist what additional experiments to perform in order to provide data that can be used for generating a model independently of the effect of monomial ordering selection, or, alternatively, generate extra simulated data points with the same purpose.

Chapter 5

Discretization of Time Course Data

5.1 Introducton

Discretization of real data into a typically small number of finite values is often required by machine learning algorithms [50], data mining [51], discrete dynamic Bayesian network applications [52], and any modeling algorithm using discrete-state models. Binary discretizations are the simplest way of discretizing data, used, for instance, for the construction of Boolean network models for gene regulatory networks [53, 54]. The expression data are discretized into only two qualitative states as either present or absent. An obvious drawback of binary discretization is that labeling the real-valued data according to a present/absent scheme generally causes the loss of a large amount of information. Discrete models and modeling techniques allowing multiple states have been developed and studied in, e.g., [47, 55]. In order to place the further discussion in a general context we give a definition of discretization [19]:

Definition 5.1.1 *A discretization of a real-valued vector* $\mathbf{v} = (v_1, \ldots, v_N)$ *is an integer-valued vector* $\mathbf{d} = (d_1, \ldots, d_N)$ *with the following properties:*

1. *Each element of* $\mathbf{d}$ *is in the set* $0, 1, \ldots, D-1$ *for some (usually small) positive integer* D*, called the degree of the discretization.*

2. *For all* $1 \leq i, j \leq N$*, we have* $d_i \leq d_j$ *if and only if* $v_i \leq v_j$*.*

Without loss of generality, assume that $\mathbf{v}$ is sorted, *i.e.* for all $i < j$, $v_i \leq v_j$. Spanning discretizations of degree D are a special case that we consider in the current work. They are defined in [19] as discretizations that satisfy the additional property that the smallest element of $\mathbf{d}$ is equal to 0 and that the largest element of $\mathbf{d}$ is equal to $D-1$. Given $\mathbf{v} = (v_1, \ldots, v_N)$, there is a large variety of scheme to obtain a discretization that is consistent with the above definition. Here we present two simple ways that are often used as a starting point in more complicated methods.

Equal Interval Width (EIW) divides the interval $[v_1, v_N]$ into k equal sized bins, where k is user-defined. Another simple method is *Equal Frequency Intervals* (EFI) which places N/k (possibly duplicated) values in each bin [50]. Any method based on those two approaches would suffer from problems that make it inapplicable to the type of biological data we work with. EIW is very sensitive to outliers and may produce a strongly skewed range [56]. In addition, some discretization levels may not be represented at all which may cause difficulties with their interpretation as part of the state space of a discrete model. In fact, for the method that we propose, we assume that for each integer a with $0 \leq a \leq D-1$, there is an entry d_i of $\mathbf{d}$ such that $a = d_i$.

On the other hand, EFI depends only on the ordering of the observed values of $\mathbf{v}$ and not on the relative spacing values. Since distance between the data points is often the only information that comes with short time courses, losing it is very undesirable. A shortcoming, common for both EIW and EFI, as well as for most other discretization methods, is that they require the number of discrete states, k, to be user-provided. We discuss later why this is impractical.

A number of entropy-based discretization methods deserve attention. An example of those is Hartemink's *Information-preserving Discretization* (IPD). It relies on minimizing the loss of pairwise mutual information between each two real-valued vectors (variables). The mutual information between two random variables X and Y with joint distribution $p(X,Y)$ and marginal distributions $p(x)$ and $p(y)$ is defined as

$$I(X;Y) = \sum_x \sum_y p(x,y) log \frac{p(x,y)}{p(x)p(y)}.$$

Note that if X and Y are independent, by definition of independence $p(x,y) = p(x)p(y)$, so $I(X;Y) = 0$. When modeling regulatory networks and having as

variables, for instance, mRNA, protein, and metabolite concentrations, the joint distribution function is rarely known and it is often hard to determine whether two variables are independent or not. In fact finding these answers is a primary reason for regulatory network modeling. Therefore, computing mutual information and basing discretization on its pairwise minimization is inapplicable.

As pointed out earlier, a major challenge of discretizing biological data is the small number of data points. For example, about 80% of microarray time series experiments are short: 3–8 time points [57]. For the case of such small samples of data, many statistical methods for discretization, such as [58], are not applicable due to the insufficient amount of the data. For example, the sample size may be insufficient to estimate distributions. We acknowledge this problem and present a method that is specifically designed to work with a small number of data points and does not make ungrounded assumptions about their statistical properties.

Another common discretization technique is based on *clustering* [59]. One of the most often used clustering algorithms is the *k-means* developed by [60]. It is a non-hierarchical clustering procedure whose goal is to minimize dissimilarity in the elements within each cluster while maximizing this value between elements in different clusters. Many applications of the k-means clustering such as the *MultiExperiment Viewer* [61] start by taking a random partition of the elements into k clusters and computing their centroids. As a consequence, a different clustering may be obtained every time the algorithm is run. Another inconvenience is that the number k of clusters to be formed has to be specified in advance. Although there are methods for choosing "the best k" such as the one described in [62], they rely on some knowledge of the data properties that may not be available.

Another method is *single-link clustering* (SLC) with the Euclidean distance function. SLC is a divisive (top-down) hierarchical clustering that defines the distance between two clusters as the minimal distance of any two objects belonging to different clusters [59]. In the context of discretization, these objects will be the real-valued entries of the vector to be discretized, and the distance function that measures the distance between two vector entries v and w will be the one-dimensional Euclidean distance $|v - w|$. Top-down clustering algorithms start from the entire data set and iteratively split it un-

til either the degree of similarity reaches a certain threshold or every group consists of one object only. For the purpose of data analysis, it is impractical to let the clustering algorithm produce clusters containing only one real value. The iteration at which the algorithm is terminated is crucial since it determines the degree of the discretization, and one of the most important features of our discretization method is a built-in termination criterion.

SLC with the Euclidean distance function satisfies one of our major requirements: very little starting information is needed – only distances between points. In addition, being a hierarchical clustering procedure it lends itself to adjustment in case that clusters need to be split or merged. It may result, however, in a discretization where most of the points are clustered into a single partition if they happen to be relatively close to one another. This negatively affects the information content of the discrete vector (to be discussed later). Another problem with SLC is that its direct implementation takes D, the desired number of discrete states, as an input. However, we would like to choose D as small as possible, without losing information about the system dynamics and the correlation between the variables, so that an essentially arbitrary choice is unsatisfactory. These two issues were addressed by modifying the SLC algorithm: our method begins by discretizing a vector in the same way as SLC but instead of providing D as part of the input, the algorithm contains termination criteria which determine the appropriate number D. After that each discrete state is checked for information content and if it is determined that this content can be considerably increased by further discretization (to be discussed later), then the state is separated into two states in a way that may not be consistent with SLC. We point out that although the discretization method we propose uses clustering as a tool to distribute the data points among the different states, it is not the same as data clustering and pursues different objectives.

5.2 Method

The method assumes that the data to be discretized consist of one or several vectors of real-valued entries. It is appropriate for applications when there is no knowledge about distribution, range, or discretization thresholds of the data and arranges the data points into clusters only according to their relative distance with respect to each other and the resulting information content.

The algorithm employs graph theory as a tool to produce a clustering of the data and provides a termination criterion.

5.2.1 Discretization of One Vector

Even if more than one vector is to be discretized, the algorithm discretizes each vector independently and for some applications this may be sufficient. The example of such a vector to keep in mind is a time course of expression values for a single gene. If the vector contains m distinct entries, a complete weighted graph on m vertices is constructed, where a vertex represents an entry and an edge weight is the Euclidean distance between its endpoints. The discretization process starts by deleting the edge(s) of highest weight until the graph gets disconnected. If there is more than one edge labeled with the current highest weight, then all of the edges with this weight are deleted. The order in which the edges are removed leads to components, in which the distance between any two vertices is smaller than the distance between any two components, a requirement of SLC. We define the distance between two components G and H to be

$$\text{dist}(G, H) = \min\{|g - h| \mid g \in G, h \in H\}.$$

The output of the algorithm is a discretization of the vector, in which each cluster corresponds to a discrete state and the vector entries that belong to one component are discretized into the same state.

Example 5.2.1 *Suppose that vector* $\mathbf{v} = (1, 2, 7, 9, 10, 11)$ *is to be discretized. The corresponding SLC dendrogram that would be obtained by SLC algorithms, such as the Johnson's algorithm [63], is given on Fig. 5.1.*

The discretization process starts by constructing the complete weighted graph based on $\mathbf{v}$ which, for the case of Example 5.2.1, corresponds to iteration 0 of the dendrogram (Fig. 5.2). Having disconnected the graph, the next task is to determine if the obtained degree of discretization is sufficient; if not, the components need to be further disconnected in a similar manner to obtain a finer discretization. A component is further disconnected if and only if both 1. and 2. below hold:

1. The minimum vertex degree of the component is less than the number of its vertices minus 1. The contrary implies that the component is a

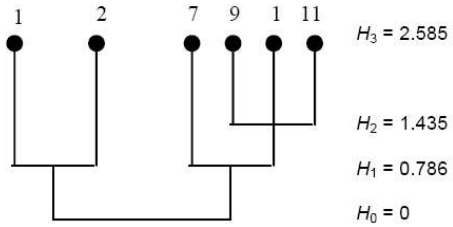


Figure 5.1: Dendrogram representing the SLC algorithm applied to the data of Example 5.2.1. The column on the right gives the corresponding Shannon's entropy increasing at each consecutive level.

complete graph by itself, *i.e.* the distance between its minimum and maximum vertices is smaller than the distance between the component and any other component.

2. One of the following three conditions is satisfied ("disconnect further" criteria):

   (a) The average edge weight of the component is greater than half the average edge weight of the complete graph.

   (b) The distance between its smallest and largest vertices is greater than or equal to half this distance in the complete graph. For the complete graph, the distance is the graph's highest weight.

   (c) Finally, if the above two conditions fail, a third one is applied: disconnect the component if it leads to a substantial increase in the information content carried by the discretized vector.

The result of applying only the first two criteria is analogous to SLC clustering with the important property that the algorithm chooses the appropriate level to terminate. Applying the third condition, the information measure criterion may, however, result in a clustering which is inconsistent with any iteration of the SLC dendrogram. This criterion is discussed in **??**.

5.2.2 Discretization of Several Vectors

Some applications may require that all vectors in a data set be discretized into the same number of states. For example the approach adopted in [47]

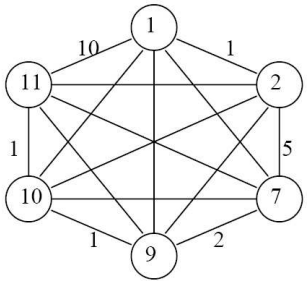


Figure 5.2: The complete weighted graph constructed from vector entries 1, 2, 7, 9, 10, 11. Only the edge weights of the outer edges are given.

imposes such a requirement on the discretization. The way we deal with this is by first discretizing all vectors separately. Suppose that for N vectors, the discretization method discretized each into $m_1, m_2, \ldots, m_N$ states, respectively. Let $m = \max\{m_i \mid i = 1, \ldots, N\}$. Now find the least possible $k = p^n$ such that $m \leq k$. Finally, discretize all variables into k states in the same way that was described for the discretization of a single vector into the required number of states.

Discretizing the entries of a real-valued vector into a finite number of states certainly reduces the information carried by the discrete vector in the sense defined in [64]. In his paper, Shannon developed a measure of how much information is produced by a discrete source. The measure is known as *entropy* or *Shannon entropy*. Suppose there is a set of n possible events whose probabilities of occurrence are known to be $p_1, p_2, \ldots, p_n$. Shannon proposed a measure of how much choice is involved in the selection of the event or how certain one can be of the outcome, which is given by

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

The base 2 of the logarithm is chosen so that the resulting units may be called bits. In our context the Shannon entropy of a vector discretized into n states is given by

$$H = \sum_{i=0}^{n-1} \frac{w_i}{n} \log_2 \frac{n}{w_i},$$

where w_i is the number of entries discretized into state i. An increase in the number of states implies an increase in entropy, with an upper bound of $\log_2 n$. However, we want the number of states to be small. That is why it is important to notice that H increases by a different amount depending on which state is split and the size of the resulting new states. For example, if a state containing the most entries is split into two new states of equal size, H will increase more than if a state of fewer entries is split or if we split the larger state into two states of different sizes.

To see that splitting a given state into two states of equal size results in maximum entropy increase, consider a vector whose entries have been divided into n states, one of which, labeled with 0, contains w_0 entries. As a function of w_0, the entropy is given by

$$H(w_0) = \frac{w_0}{n} \log_2 \frac{n}{w_0} + \sum_{i=0}^{n-1} \frac{w_i}{n} \log_2 \frac{n}{w_i}. \tag{5.1}$$

Suppose that we split state 0 into two states containing m and $w_0 - m$ entries, respectively, where $0 < m < w_0$. This will change only the first term of the right-hand side of (5.1) and leave the summation part the same. It is easy to verify that $h(w_0) = \frac{w_0}{n} \log_2 \frac{n}{w_0}$ achieves its maximum value over $0 < m < w_0$ at $m = w_0/2$. Therefore, splitting a state into two states of equal size maximizes the entropy increase.

As explained in the previous section, the information measure criterion is applied to a component only after the component has failed the other three conditions. Once this happens, we consider splitting it further only if doing so would provide a very significant increase of the entropy, *i.e.* if the component corresponds to a "large" collection of entries (recurring entries are included since all entries have to be considered when computing the information content of a vector). In our implementation *a component gets disconnected further only if it contains at least half the vector entries.* Unlike with the other criteria, if a component is to be discretized under the information condition, the corresponding sorted entries are split into two parts: not between the two most distant entries but into two equal parts (or with a difference of one entry in case of an odd number of entries). This is to guarantee a maximum increase of the information measure.

In Example 5.2.1, the two components that were obtained by removing the edges of heaviest weight both fail the "disconnect further" conditions. If

the discretization process stopped at this iteration, vector $\mathbf{d} = (0, 0, 1, 1, 1, 1)$ would have Shannon entropy 0.78631. Having most of the entries of $\mathbf{v}$ discretized into the same state, 1, reduces the information content of $\mathbf{d}$.

Suppose discretization of $\mathbf{v}$ continues according to SLC, *i.e.*, without enforcing the fourth condition of "disconnect further". The next step is to remove the edges of highest weight until a component gets disconnected. This yields the removal of the four edges of weights 4, 3, 2, and 2, respectively, to obtain $d = (0, 0, 1, 2, 2, 2)$. The Shannon entropy of the new discretization of $\mathbf{v}$ is 1.43534. Still half of the entries of $\mathbf{v}$ remain at the same discrete level, now 2, which does not allow for a maximal increase in the information content of $\mathbf{d}$. If instead discretization proceeded by applying the information criterion to the bigger component, the resulting discretization becomes $d = (0, 0, 1, 1, 2, 2)$ with Shannon entropy 1.58631, as opposed to the previous entropy of 1.43534.

As illustrated by the example of discretizing vector $v = (1, 2, 7, 9, 10, 11)$, the proposed discretization algorithm produces a discretization which is consistent with the definition given above, keeps the number of discrete states small, and maximizes information content over traditional SLC.

5.3 Algorithm Summary

An implementation of the algorithm is available from the authors at `http://polymath.vbi.vt.edu/discretization`.

Input: set $S_r = \{v_i \mid i = 1, \ldots, m\}$ where each $v_i = (v_{i1}, \ldots, v_{iN})$ is a real-valued vector of length N to be discretized.

Output: set $S_d = \{d_i \mid i = 1, \ldots, m\}$ where each $d_i = (d_{i1}, \ldots, d_{iN})$ is the discretization of v_i for all $i = 1, \ldots, m$.

1. For each $i = 1, \ldots, m$, construct a complete weighted graph G_i where each vertex represents a distinct v_{ij} and the weight of each edge is the Euclidean distance between the incident vertices.

2. Remove the edge(s) of highest weight.

3. If G_i is disconnected into components $C_{i1}^{G_i}, \ldots, C_{iM_i}^{G_i}$, go to 4. Else, go to 2.

4. For each $C_{ik}^{G_i}$, $k = 1, \ldots, M_i$, apply "disconnect further" criteria 1-3. If any of the three criteria holds, set $G_i = C_{i1}^{G_i}$ and go to 2. Else, go to 5.

5. Apply "disconnect further" 2c. If this criterion is satisfied, go to 6. Else, go to 7.

6. Sort the vertex values of $C_{i1}^{G_i}$ and split them into two sets: if $|V(C_{i1}^{G_i})|$ is even, split the first $|V(C_{i1}^{G_i})|/2$ sorted vertex values of $G_i = C_{i1}^{G_i}$ into one set and the rest – into another. If $|V(C_{i1}^{G_i})|$ is odd, split the first $|V(C_{i1}^{G_i})|/2 + 1$ sorted vertex values of $|V(C_{i1}^{G_i})|$ into one set and the rest – into another.

7. Sort the components $C_{i1}^{G_i}$, $k = 1, \ldots, M_i$, by the smallest vertex value in each $C_{i1}^{G_i}$ and enumerate them $0, \ldots, D_i - 1$, where D_i is the number of components into which G_i got disconnected. For each $j = 1, \ldots, N$, d_{ij} is equal to the label of the component in which v_{ij} is a vertex.

5.4 Algorithm Complexity

Given M variables, with N time points each, we compute $N(N-1)/2$ distances to construct the distance matrix so the complexity of this step is $\mathcal{O}(N^2)$. The distance matrix is used to create the edge and vertex sets of the complete distance graph, containing $N(N-1)/2$ edges. This can also be accomplished in $\mathcal{O}(N^2)$ time. These edges are then sorted in decreasing order, so that the largest edges are removed first. A standard sorting algorithm, such as merge sort, has complexity $\mathcal{O}(N \log N)$ [65]. As each edge is removed, the check for graph disconnection involves testing for the existence of a path between the two vertices of the edge. This test for graph disconnection can be accomplished with a breadth-first search, which has order $\mathcal{O}(E+V)$ [66], with E the number of edges and V the number of vertices in the component. In our case this translates to complexity $\mathcal{O}(N^2)$. Edge removal is typically performed for a large percentage of the $N(N-1)/2$ edges, so this step has overall complexity $\mathcal{O}(N^4)$. The edge removal step dominates the complexity so that the overall complexity is $\mathcal{O}(MN^4)$ to discretize all M variables. While

this is the theoretical worst-case performance, because of the heuristics we have added the typical performance is significantly better.

5.5 Inconsistencies in the Discretized Data

One of our objectives is to introduce a discretization method suitable specifically for time courses of data. The algorithm is more general and can be applied to any collection of data points disregarding their order. If, however, the time courses are to be fitted by a deterministic dynamical system, as is the case with most reverse engineering methods, the order of the points becomes important. If any of the input time courses contain consecutive points $(d_1, \ldots, d_N) \to (a_1, \ldots, a_N)$ and $(d_1, \ldots, d_N) \to (b_1, \ldots, b_N)$, then we should require that $a_j = b_j$ for all $j = 1, \ldots, N$. Even if the experimental data satisfy this requirement, the discretization may create inconsistencies. To obtain a usable time course while at the same time discretizing consistently, several approaches are possible. One is to coarsen the discretization by merging discrete states. Although this would certainly solve the inconsistency problem, it would also reduce the information content of the discrete data. The approach that we adopt is to split as many times as necessary the discrete states of the problematic point $(d_1, \ldots, d_N)$. The state to split first is naturally the one containing the two most distant consecutive real values. Assuming that the analog time course data are consistent, discretization-induced inconsistencies are handled in the following way:

In the discretized time course, find all points $(d_1, \ldots, d_N)$ for which there are at least two distinct points $(a_1, \ldots, a_N)$ and $(b_1, \ldots, b_N)$ that immediately follow point $(d_1, \ldots, d_N)$ anywhere in the time course. For each $(d_1, \ldots, d_N)$:

1. For each d_j find all real values $x_{j,i}$ that were discretized into state d_j, sort them, and re-index them.

2. Let $d_j^{\max} = \max\{|x_{j,i+1} - x_{j,i}|\}$ and let $(x_{j,\text{left}}, x_{j,\text{right}}) = (x_{j,i+1} - x_j)$ for which $|x_{j,i-1} - x_{j,i}| = d_j^{\max}$.

3. Let $D_{\max} = \{d_j^{\max} \mid j = 1, \ldots, N\}$. Split state d_j that contains values $x_{j,\text{left}}$ and x_j, right for which $x_{j,\text{right}} - x_{j,\text{left}} = D_j^{\max}$. Re-label states accordingly.

Repeat until no inconsistencies are present.

5.6 Requirements on the Number of States

While for some applications any number of discretization states is acceptable, there are some cases when there are limitations on this number. For example, if the purpose of discretizing the data is to build a model of polynomials over a finite field as in [47], then the number of states must be a power of a prime p^n since every finite field has such a cardinality. Our method deals with this problem in the following way.

Suppose that a vector has been discretized into m states in the way described above. The next step is to find the smallest integer $k = p^n$ such that $m \leq k$. This value for k gives the number of states that needs to be obtained. Since the discretization algorithm yielded m clusters, 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 with respect to Euclidean distance. The splitting should take place between these entries. This is repeated until k clusters are obtained.

This approach has a potential problem. For instance, if a vector got discretized into 14 states and the total number of distinct entries of the vector is 15, then $k = 16$ cannot be reached. In this case the two closest states could be merged together to obtain 13 states. In general it may not be desirable to reduce the number of states because this results in loss of information. We choose to increase the number of states unless it is impossible, as in the above example.

5.7 Preservation of Dynamics

As mentioned above, the discretization algorithm is designed to preserve the dynamic features of time course data. Due to limited knowledge of the dynamic features of real biochemical networks, the validation of our method is best done with a simulated network, as demonstrated in Section 6.

5.8 Discretization in the Presence of Noise

In the example from Section 6 no noise is added to the time courses generated from the artificial gene network. Noise, however, is naturally present in biological data and in microarray data in particular [70]. Although there are

various techniques which increase the accuracy of the microarray measurements, the data inevitably contain errors due to the probabilistic characteristics of the detection process, from sample extraction and mRNA purification to hybridization and imaging. Consequently, it is crucial to study how the proposed discretization algorithm performs in the presence of typical levels of noise in the experimental data.

5.8.1 Noiseless data

To study the effect of noise, we considered two types of data. We used gene expression measurements generated for a study of rat cervical central nervous system development [71]. The data consist of nine expression measurements for each gene: cervical spinal cord tissue was dissected from animals in embryonic development at days 11, 13, 15, 18, and 21 and at postnatal days 0, 7, 14, and 90. We selected 18 genes whose nine-point time courses have statistical variance of more than 1. Although the data likely contain some noise, here we assume that the measurements are perfect and we add noise of known proportion and distribution assuming it is the only noise in the data. The purpose is to work with data which have the properties of real microarray time courses. The second data set is an artificial ten-point time course on 30 "genes". The values are randomly generated real numbers in the range $[0, 20]$ with statistical variance greater than 10.

5.8.2 Adding noise to the data

Two types of noise are added to the data at the same time: overall and point-specific [72]. The overall noise is added by sampling from a normal distribution with zero mean and a standard deviation equal to 12.5% of the standard deviation of each gene. The point-specific noise is simulated by adding noise to each time point sampling from a different normal distribution with zero mean and standard deviation equal to 12.5% of the standard deviation of the particular time point value. Thus, the total proportion of added noise amounts to 25%. For each gene 100 noise containing replicates were generated and discretized.

5.8.3 Results

For each gene, we compare the way the original noiseless time course was discretized to the discretization of the noise-containing replicates. We count the number of noise-containing replicates that got discretized exactly the same as the original noiseless time course. For the gene expression data from [71], 78.72% of all the 18×100 noise-containing replicates were discretized exactly as their corresponding noiseless original. This number is significantly higher for the randomly generated data of variance 10 or higher: 93.8% of all the 30×100 noise-containing replicates were discretized exactly as their corresponding noiseless original. The big difference between the two results can be explained by the different statistical variance of the original data in the two cases. Only 5 out of the 18 genes in the expression measurements time course in [71] have variance greater than 10 while the "genes" in the simulated time course had variance of at least 10 with average variance of 34.5. Since the higher the variance of a time course, the better the chance of more distinct and easier to detect discrete states, the discretization of the data with higher variance not surprisingly showed more robustness in the presence of noise.

5.9 Conclusion

The method presented is particularly suitable for the discretization of short multivariate time courses, since it preserves a large degree of information about dynamic features and ensures data consistency, without making ungrounded assumptions about the data and their source. It thus provides a valuable tool for any application that requires discretization of continuous data when the number of discrete classes that best fits the data is unknown. An important advantage of using the discrete states determined by the method is that a significant portion of the noise is absorbed in the process.

Chapter 6

Example: Data Discretization and Reverse Engineering of a Simulated Gene Regulatory Network

In this section we demonstrate how the data discretization method presented in Section 5 can be applied for the reconstruction of a gene network. Since data from real gene networks are limited, we chose to test the methods on an artificial gene network. We used the *A-Biochem* software system developed by P. Mendes and his collaborators [67]. *A-Biochem* automatically generates artificial gene networks with particular topological and kinetic properties. These networks are embodied in kinetic models, which are used by the biochemical network simulator *Gepasi* [69] to produce simulated gene expression data. We generated an artificial gene network with five genes $G_1, \ldots, G_5$ and ten total input connections using the Albert-Barabási algorithm [68]. The relationships among the genes are given in Table 6.1. *Gepasi* uses a continuous representation of biochemical reactions, based on ODEs. With the parameters we specified, *Gepasi* generated the following ODE system that

Gene	Activator	Inhibitor
G_1	G_1	G_3
G_2	G_1	G_3
G_3	–	G_1, G_3, G_5
G_4	–	G_3
G_5	G_1, G_3	–

Table 6.1: Relationships among the five genes of the *A-Biochem*-generated artificial gene network.

represents the network:

$$\begin{aligned}
\frac{\mathrm{d}G_1}{\mathrm{d}t} &= \frac{0.01\left(1+\frac{G_1(t)}{0.01+G_1(t)}\right)}{0.01+G_3(t)} - G_1(t) \\
\frac{\mathrm{d}G_2}{\mathrm{d}t} &= \frac{0.01\left(1+\frac{G_1(t)}{0.01+G_1(t)}\right)}{0.01+G_3(t)} - G_2(t) \\
\frac{\mathrm{d}G_3}{\mathrm{d}t} &= \frac{10^{-6}}{(0.01+G_1(t))(0.01+G_3(t))(0.01+G_5(t))} - G_3(t) \qquad (6.1) \\
\frac{\mathrm{d}G_4}{\mathrm{d}t} &= \frac{0.01}{0.01+G_3(t)} - G_4(t) \\
\frac{\mathrm{d}G_5}{\mathrm{d}t} &= \left(1+\frac{G_1(t)}{0.01+G_1(t)}\right)\left(1+\frac{G_3(t)}{0.01+G_3(t)}\right) - G_5(t)
\end{aligned}$$

Analyzing the dynamics of the ODE system, one finds that it has two stable steady states (of which only the first is bio-chemically meaningful):

$$S_1 = (1.99006, 1.99006, 0.000024814, 0.997525, 1.99994)$$

and

$$S_2 = (-0.00493694, -0.00493694, -0.0604538, -0.198201, 0.0547545).$$

As [47] demonstrated, the performance of their algorithm dramatically improves if *knockout* time courses for genes are incorporated. Genetic knockout is the process of replacing a specific gene with an inactive or mutated allele. For this reason we supplied seven time course of 11 points each: two wild-type time courses and five knockout time courses, one for each gene. The first

wild-type time course is generated by solving the ODE system numerically for $t = 0, 2, 6, \ldots, 20$ with initial conditions $G_i(0) = 1$ for all $i = 1, \ldots, 5$. Fig. 6.1 shows a plot of the numerical solution of the ODE system with these initial conditions.

The second time series is generated like the first one but this time with $t = 0, 1, \ldots, 10$ and initial conditions $(G_1(0), G_2(0), G_3(0), G_4(0), G_5(0)) = (1, -1, -0.6, -1, 0.5)$. (We emphasize that we are including the steady state S_2 in order to show that the discretized data preserve information about the dynamics of the ODE system.) One can simulate a gene knockout in *Gepasi* by setting the corresponding variable and initial condition to zero. In this case, the time points from each of the seven time courses constitute the input vectors. The discretization algorithm chose a state set of cardinality 5 and, based on the discrete data, the reverse engineering method of [47] generated the discrete model $F = (f_1, f_2, f_3, f_4, f_5) : \mathbb{F}_5^5 \to \mathbb{F}_5^5$ with coordinate polynomials

$$\begin{aligned}
f_1 &= 2x_5^2 2x_5 x_1^2 2x_2 x_1^2 + 2x_5 x_3^2 + 2x_5 x_1 x_2 x_1 2x_5 x_3 + x_5 + x_2 \\
&\quad 2x_1 x_3^2 + 2x_1^2 + x_1 x_3 2x_3^2 + 2x_3 \\
f_2 &= 2x_5^2 2x_5 x_3^2 + x_5 x_1 2x_2 x_1 + x_5 x_3 + x_5 + x_2 + x_1^4 x_1 x_3^2 x_3^3 \\
&\quad 2x_1^2 x_1 x_3 + 2x_3^2 + x_1 + x_3 \\
f_3 &= x_3 x_1^2 + 2x_3 x_1 x_5 x_3 x_5^2 x_3 \\
f_4 &= 2x_2 x_1 2x_1 x_3^2 2x_2 x_4 + x_1 x_4 + x_2 x_3 + 2x_1 x_3 2x_2 x_1 \\
&\quad x_4^3 + x_3^3 x_3^2 + 2x_4 \\
f_5 &= 2x_4 x_5 x_1 2x_4 x_1^2 2x_4 x_3^2 x_4 x_5 x_4 x_3 2x_4 + 2x_5^2 x_1 x_1^3 + 2x_1 x_3^2 \\
&\quad x_3^3 x_5^2 2x_5 x_1 x_1^2 + x_1 x_3 + x_5 x_3 2
\end{aligned} \tag{6.2}$$

Now we compare the dynamics of the two models. First, the discretization maps steady state S_1 of the ODE system to the fixed point $FP_1 = (4, 4, 1, 4, 2)$ of F and steady state S_2 to the fixed point $FP_2 = (0, 1, 1, 1, 0)$ of F. The time course produced by solving the ODE system and converging to S_1 is given in the top part of Fig. 6.2.

The corresponding discrete points from the time course in the bottom part of Fig. 6.2 form a trajectory that ends at FP_1 (Fig. 6.3). The discrete model trajectory can be superimposed over the discretization of the continuous one, illustrating the matching dynamics of the two models. The same can be observed for the second steady-state S_2 that is mapped to fixed point FP_2.

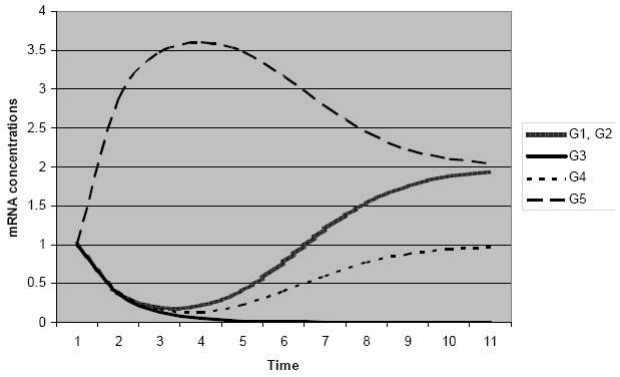
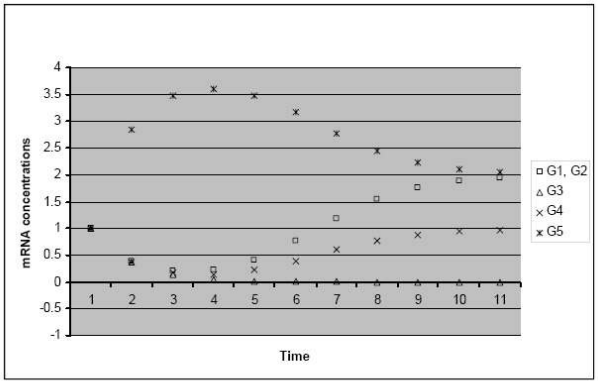


Figure 6.1: Plot of the numerical solution of (6.1) with initial condition $(G_1(0), G_2(0), G_3(0), G_4(0), G_5(0)) = (1, 1, 1, 1, 1)$.

Further evidence of the ability of our method to retain information in the data is the fact that the reverse-engineering method in [47] can extract most of the information about the wiring diagram of the network. This can be seen by comparing the inferred diagram with the diagram of actual direct interactions in Fig. 6.4.



Time	G_1	G_2	G_3	G_4	G_5
0	3	3	4	4	1
2	1	1	3	2	4
4	1	1	2	2	4
6	1	2	1	2	4
8	3	3	1	3	3
10	4	4	1	4	2
12	4	4	1	4	2
14	4	4	1	4	2
16	4	4	1	4	2
18	4	4	1	4	2
20	4	4	1	4	2

Figure 6.2: Top: Wild-type time course generated by solving numerically the ODE system (6.1) for $t = 0, \ldots, 10$ with initial conditions $(G_1(0), G_2(0), G_3(0), G_4(0), G_5(0)) = (1, 1, 1, 1, 1)$. Bottom: Corresponding discrete point time course.

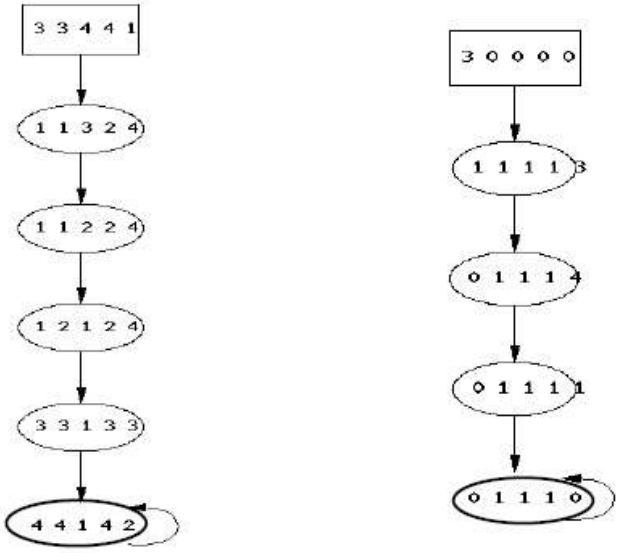


Figure 6.3: Trajectories formed by the discretized wild-type time courses [73].

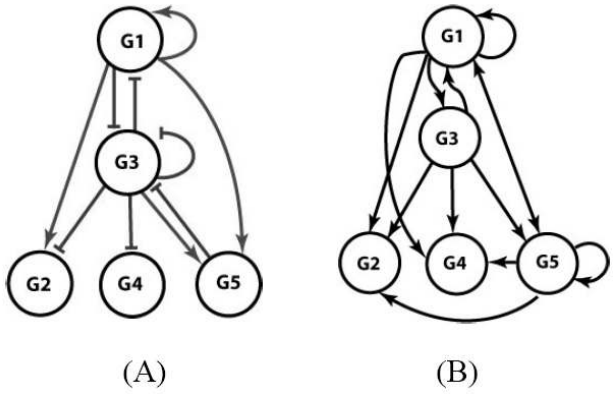


Figure 6.4: (A) Wiring diagram of (6.1); (B) Wiring diagram of (6.2).

Chapter 7

Discussion and Future Work

In this work we considered two aspects of polynomial modeling over finite fields for systems biology: dependence on monomial ordering and data discretization. We propose techniques to systematically deal with these issues which improve the quality of the model, making it independent of the specifics of Gröbner basis computation and increasing the amount of information extracted from the available data.

The relationship between points in k^n and the number of Gröbner bases of their corresponding ideal of points is of interest by itself even outside of the context of any modeling application. Nevertheless we emphasize that our main goal was similar to that of [74]: to measure the amount of data necessary for obtaining a correct polynomial model generated with the method in [47]. The model "correctness" we addressed, however, is not measured directly by how well the model reflects the actual network but rather by how few unsupported by the data assumptions we make while generating the model. Since choosing a monomial ordering is most often such an assumption, eliminating the necessity for it improves the model correctness in this sense.

Especially for the application of the method to the reverse engineering algorithm in [47], we would like to investigate how the normal form of a given polynomial model changes when taken with respect to different reduced Gröbner bases. A powerful tool in this research would be the Gröbner fan of the ideal of points (see Section 3.1.2) in which every cone corresponds to a marked reduced Gröbner basis. It would probably be beneficial to study how the position of the cones on the Gröbner fan are related to the normal

form of the polynomial model under consideration.

The data discretization method presented in this dissertation has several novel features. It uses Shannon's information criterion to determine clusters, identifies the optimal number of clusters for a given data set, and eliminates inconsistencies in the discrete time courses. It is particularly suitable for the discretization of multivariate time courses, since it preserves a large degree of information about dynamic features and ensures data consistency. It thus provides a valuable tool for any application that requires discretization of continuous data when the number of discrete classes that best fits the data is unknown. An important advantage of using discrete states is that a significant portion of the noise is absorbed in the process. The experiments we carried out make us confident that for data that discretize into a relatively small number of states and that contain a degree of noise common to many biological data, the majority of the noise is absorbed into the discrete states. We do not, however, recommend using the method whenever a large amount of data is available. In this case, a statistical method, such as [58], that can take advantage of the statistical properties of the data may be more appropriate. Our method assumes no knowledge of these properties and therefore cannot utilize this information. Another situation when virtually any discretization technique may not produce useful results is in the case when one or several of the system variables change much more rapidly than the rest. In light of the discussion in Section 5.5 it is clear that in order to avoid inconsistencies in the discretized data, one may need a very large number of discrete states which may be a disadvantage for the modeling process.

Bibliography

[1] Whitmarsh, J. (2005). The Need for Mathematics in Biomedical Research. Presentation in the Joint Mathematics Meeting of the AMS and the MAA, AMS Special Session on Mathematical Sciences Contributions to the Biomedical Sciences II.

[2] von Bertalanffy, L. (1968). *General System Theory: Foundations, Development, Applications*, New York: George Braziller.

[3] Wiener, N. (1948). *Cybernetics*, Cambridge, MA:MIT Press.

[4] Kitano, H. (2002). Systems Biology: a Brief Overview. *Science*, **295**, pp. 1662–1664.

[5] Vidal, M., Furlong, E.E.M. (2004). From OMICS to Systems Biology. *Nature Reviews*, Genetics 2004.

[6] Percus, J. (2001). *Mathematics of Genome Analysis*, Cambridge University Press.

[7] Murray, J.M. (2001). *Mathematical Biology*, Springer-Verlag.

[8] Wolkenhauer, O. (2005). *Mathematical Systems Biology*, manuscript.

[9] Cohen, J.E. (2004). Mathematics Is Biology's Next Microscope, Only Better; Biology Is Mathematics' Next Physics, Only Better. *PLoS Biol* 2(12): e439.

[10] Sha, W. (2006). Microarray Data Analysis Methods and Their Applications to Gene Expression Data Analysis for *Saccharomyces Cerevisiae* under Oxidative Stress. Ph. D. Dissertation, Virginia Polytechnic Institute and State University.

[11] Christen, Y. (2000) Oxidative Stress and Alzheimer Disease. *Am. J. Clin. Nutr.*, **71**, pp. 621S– 629S.

[12] Maritim, A.C., Sanders, R.A., Watkins, J.B., 3rd (2003). Diabetes, Oxidative Stress, and Antioxidants: a Review. *J. Biochem. Mol. Toxicol.*, **17**, pp. 24–38.

[13] Klaunig, J.E., Kamendulis, L.M. (2004). The Role of Oxidative Stress in Carcinogenesis. *Annu. Rev. Pharmacol. Toxicol.*, **44**, pp. 239–267.

[14] Yu, B.P. (1994). Cellular Defenses Against Damage from Reactive Oxygen Species. *Physiol. Rev.*, **74**, pp. 139–162.

[15] Jamieson, D.J. (1998). Oxidative Stress Responses of the Yeast *Saccharomyces cerevisiae*, *Yeast*, **14**, pp. 1511–1527.

[16] Mendes, P., Kell, D.B. (1998). Non-linear Optimization of Biochemical Pathways: Applications to Metabolic Engineering and Parameter Estimation. *Bioinformatics*, **14**, pp. 869–883.

[17] Briggs, G.E., Haldane, J.B.S. (1925). A Note on the Kinetics of Enzyme Action, *Biochem. J.*, **19**, pp. 339–339.

[18] Yeung, M.K.S., Tegnér, J., Collins, J.J. (2002). Reverse Engineering Gene Networks Using Singular Value Decomposition and Robust Regression. *Proc. Natl. Acad. Sci.*, **99**, pp. 6163–6168.

[19] Hartemink, A. (2001). Principled Computational Methods for the Validation and Discovery of Genetic Regulatory Networks. Ph. D. dissertation, Massachusetts Institute of Technology.

[20] Filkov V., Istrail, S. (2002). Inferring Gene Transcription Networks: the Davidson Model. *Genome Informatics*, **13**, pp. 236–239.

[21] Lewis, J.E., Glass, L. (1991). Steady States, Limit Cycles, and Chaos in Models of Complex Biological Networks. *Int. J. Bifurcation and Chaos*, **1**, pp. 477–483.

[22] Kauffman, S.A. (1993). *The Origins of Order: Self-organization and Selection in Evolution*. Oxford University Press, New York.

[23] Friedman, N., Linial, M., Nachman, I., Pe'er, D. (2000). Using Bayesian Networks to Analyze Expression Data. In *4th Annual International Conference on Computational Molecular Biology (RECOMB 2000)*, ACM-SIGACT.

[24] von Neumann, J. (1966). *The Theory of Self-reproducing Automata*, A. Burks, ed., Univ. of Illinois Press, Urbana, IL.

[25] Gardner, M. (1970). The Fantastic Combinations of John Conway's New Solitaire Game "Life". *Scientific American*, October 1970.

[26] Wolfram, S. (1982). Cellular Automata as Simple Self-Organizing Systems. Caltech preprint CALT-68-938.

[27] Weimar, J., Boon, J. (1994). Class of Cellular Automata for Reaction-diffusion Systems. *Phys. Rev. E*, **49**(2), pp. 1749-1752.

[28] Alber, M., Kiskowski, M., Glazier, J., Jiang, Y. (2002). On Cellular Automaton Approaches to Modeling Biological Cells, *IMA 134: Mathematical Systems Theory in Biology, Communication, and Finance*, p. 12, Springer-Verlag, New York.

[29] Thomas, R., D'Ari, R. (1990). *Biological Feedback*, CRC Press, Boca Raton, Ann Arbor, Boston.

[30] Thomas, R. (1973). Boolean Formalization of Genetic Control Circuits. *J. Theor. Biol.*, **42**, p. 563.

[31] Mendoza, L., Thieffry, D., Alvarez-Buylla, E.R. (1999). Genetic Control of Flower Morphogenesis in *Arabidopsis thaliana*: a logical analysis. *Bioinformatics*, **15**, pp. 593–606.

[32] Sanchez, L., Thieffry, D. (2001). A logical Analysis of the *Drosophila* Gap Genes. *J. Theor. Biol.*, **211**(2), pp. 115–141.

[33] Albert, R. (2004). Boolean Modeling of Genetic Regulatory Networks. In: *Complex Networks*, Editors: Ben-Naim, E., Frauenfelder, H., Toroczkai, Z., Springer-Verlag.

[34] Möller, M. (1998), Gröbner bases and numerical analysis. In Buchberger, B., Winkler, F. eds, *Gröbner Bases and Applications (Proceedings of the Conference on 33 Years of Gröbner Bases)*, volume 251 of *London Mathematical Society Lecture Notes Series*, pp. 159–178. Cambridge University Press.

[35] Sakata, S. (1998). Gröbner bases and coding theory. In Buchberger, B., Winkler, F. eds, *Gröbner Bases and Applications (Proceedings of the Conference 33 Years of Gröbner Bases)*, volume 251 of *London Mathematical Society Lecture Notes Series*, pp. 205–220. Cambridge University Press.

[36] Robbiano, L. (1998). Gröbner bases and statistic. In Buchberger, B., Winkler, F. eds, *Gröbner Bases and Applications (Proceedings of the Conference 33 Years of Gröbner Bases)*, volume 251 of *London Mathematical Society Lecture Notes Series*, pp. 179–204. Cambridge University Press.

[37] Buchberger, B. (1965). An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-dimensional Polynomial Ideal. Ph.D. Thesis, University of Innsbruck.

[38] Buchberger, B., Möller, H. M. (1982). The construction of multivariate polynomials with preassigned zeros. In Calmet, J. ed., *Proceedings of the European Computer Algebra Conference (EUROCAM'82) (Marseille, France)*, LNCS **144**, pp. 24–31. Springer.

[39] Abbott, J., Bigatti, A., Kreuzer, M., Robbiano, L. (2000). Computing Ideals of Points. *J. Symbolic Computation*, **30**, pp. 341–356

[40] Cox, D., Little, J., and O'Shea, D. (1997). *Ideals, Varieties, and Algorithms*, Springer-Verlag, New York.

[41] Adams, W. W., Loustaunau, P. (1994). *An Introduction to Gröbner Bases*, Americal Mathematical Society, Graduate Studies in Math. Vol. III.

[42] Eisenbud, D. (1995). *Introduction to Commutative Algebra with a View Towards Algebraic Geometry*, Graduate Texts in Mathematics, Springer, New York.

[43] Buchberger, B. (1985). In Bose, N. K. ed., *Gröbner Bases: an Algorithmic Method in Polynomial Ideal Theory*, chapter 6, pp. 184–232. D. Reidel Publishing Co.

[44] Mora, T., Robbiano, L. (1988). Gröbner Fan of an Ideal. *J. Symbolic Computation*, **6**(2/3), pp. 183–208.

[45] Jensen, A. N. (2005). Gfan, a Software System for Gröbner Fans. Available at `http://home.imf.au.dk/ajensen/software/gfan/gfan.html`.

[46] Bayer, D., Stillman, M. (1987). A Theorem on Refining Division Orders by the Reverse Lexicographic Order. *Duke J. Math*, **55**, pp. 321–328.

[47] Laubenbacher, R., Stigler, B. (2004). A computational algebra approach to the reverse engineering of gene regulatory net-works. *J. Theor. Biol.* **229**, pp. 523–537.

[48] Lidl, R., Niederreiter, H. (1997). *Finite Fields*, 2nd ed., Encyclopedia of Mathematics and Its Applications, vol. 20, Cambridge University Press, New York.

[49] Stigler, B. (2005). An Algebraic Approach to Reverse Engineering with an Application to Biochemical Networks. Ph. D. Dissertation, Virginia Polytechnic Institute and State University.

[50] Dougherty J., Kohavi R., Sahami M. (1995). Supervised and Unsupervised Discretization of Continuous Features. In *Machine learning: Proceedings of the 12th International Conference*, San Francisco, CA. In Prieditis, A. and Russell, S. (eds.): Morgan Kauffman.

[51] Han J., Kamber M. (2000). *Data Mining: Concepts and Techniques*, Academic Press, San Diego, CA.

[52] van Berlo R., van Someren E., Reinders M. (2003). Studying the Conditions for Learning Dynamic Bayesian Networks to Discover Genetic Regulatory Networks. *SIMULATION*, **79**(12), pp. 689–702.

[53] Kauffman S.A. (1969). Metabolic Stability and Epigenesist in Randomly Constructed Genetic Nets. *J. Theor. Biol.*, **22**, pp. 437–467.

[54] Albert R., Othmer H. (2003). The Topology of the Regulatory Interactions Predict the Expression Pattern of the Segment Polarity Genes in *Drosophila melanogaster. J. Theor. Biol.*, **223**, pp. 1–18.

[55] Thieffry D., Thomas R. (1998) Qualitative Analysis of Gene Networks. In *Proc. Pacific Symp. on Biocomputing*, Singapore. World Scientific, pp. 77–88.

[56] Catlett, J. (1991). Megainduction: Machine Learning on Very Large Databases, Ph. D. dissertation, University of Sydney.

[57] Ernst, J., Bar-Joseph, Z. (2006). STEM: a Tool for the Analysis of Short Time Series Gene Expression Data. *BMC Bioinformatics*, **7**, p. 191.

[58] Pe'er, D., Regev, A., Elidan, G., Friedman, N. (2001). Inferring Subnetworks from Perturbed Expression Profiles. *Bioinformatics*, **17**,pp. S215–224.

[59] Jain, A., Dubes, R. (1988). *Algorithms for Clustering Data*, Prentice Hall.

[60] MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the 5th Berkeley Symposium of Mathematical Statistics and Probability*. Berkeley, CA. University of California Press, **1**, pp. 281–297.

[61] Saeed, A., Sharov, V., White, J., Li, J., Liang, W., Bhagabati, N., Braisted, J., Klapa, M., Currier, T., Thiagarajan, M., Sturn, A., Snuffin, M., Rezantsev, A., Popov, D., Ryltsov, A., Kostukovich, E., Borisovsky, I., Liu, Z., Vinsavich, A., Trush, V., Quackenbush, J. (2003). TM4: a Free, Open-source System for Microarray Data Management and Analysis. *BioTechniques*, **34**(2), pp. 374–378.

[62] Crescenzi, M., Giuliani, A. (2001). The Main Biological Determinants of Tumor Line Taxonomy Elucidated by of Principal Component Analysis of Microarray Data. *FEBS Letters*, **507**, pp. 114–118.

[63] Johnson, S.C. (1967). Hierarchical Clustering Schemes. *Psychometrika*, **32**, pp. 241–254.

[64] Shannon, C. (1948). A Mathematical Theory of Communication. *The Bell Systems Technical Journal*, **27**, pp. 379–423, 623–656.

[65] Knuth, D.E. (1998). *The Art of Computer Programming, Vol. 3: Sorting and Searching*, 2nd edition. Reading, Massachusetts, Addison-Wesley.

[66] Pemmaraju, S., Skiena, S. (2003). *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Cambridge University Press.

[67] Mendes, P., Sha, W., Ye, K. (2003). Artificial Gene Networks for Objective Comparison of Analysis Algorithms. *Bioinformatics*, **19**, pp. ii122–ii129.

[68] Albert, R., Barabási, A. (2000). Topology of Evolving Networks: Local Events and Universality. *Phys. Rev. Lett.*, **85**, pp. 5234–5237.

[69] Mendes, P. (1993). GEPASI: a Software Package for Modeling the Dynamics, Steady States and Control of Biochemical and Other Systems. *Comput. Appl. Biosci.*, **9**, pp. 563–571.

[70] Hassibi, A., Vikalo, H. (2005). Probabilistic Modeling and Estimation of Gene Expression Levels in Microarray. In *Proc. IEEE Workshop on Genomic Signal Processing and Statistics (GENSIPS)*.

[71] Wen, X., Fuhrman, S., Michaelis, G., Carr, D., Smith, S., Barker, J., Somogyi, R. (1998). Large-scale Temporal Gene Expression Mapping of Central Nervous System Development. *Proc. Natl. Acad. Sci. USA*, **95**, pp. 334–339.

[72] Hatzimanikatis, V., Lee, K.H. (1999). Dynamical Analysis of Gene Networks Requires Both mRNA and Protein Expression Information. *Metab. Eng.*, **1**, pp. 275–281.

[73] `http://dvd.vbi.vt.edu`

[74] Just, W. (2006). Reverse Engineering Discrete Dynamical Systems from Data Sets with Random Input Vectors. *Mathematical Biosciences Institute*, technical publication No. 52.

Vita

Elena Dimitrova was born in Sofia, Bulgaria to a family of engineers. Following high school, she attended the American University in Bulgaria, where in 2001 she received a B.A. in computer science with a minor in mathematics. She continued her education at Virginia Tech, earning a M.S. in 2003. Two years later she received a Ph.D. in mathematics for her work at the Virginia Bioinformatics Institute at Virginia Tech. She has accepted the position of Assistant Professor of Mathematical Sciences and Adjunct Professor of Genetics and Biochemistry at Clemson University. Elena is a member of Society for Industrial and Applied Mathematics, American Mathematical Society, and Society for Advancement of Chicanos and Native Americans. She is a Fellow of Project NExT, a program of the Mathematical Association of America.