INFERENCE OF SWITCHING NETWORKS BY USING A PIECEWISE LINEAR FORMULATION

DİDEM AKÇAY

DECEMBER 2005

INFERENCE OF SWITCHING NETWORKS BY USING A PIECEWISE LINEAR FORMULATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

DİDEM AKÇAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF SCIENTIFIC COMPUTING

DECEMBER 2005

Approval of the Graduate School of Applied Mathematics

Prof. Dr. Ersan AKYILDIZ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Bülent KARASÖZEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Hakan Öktem Supervisor	Prof. Dr. Semra Kocabıyık Co-supervisor

Examining Committee Members

Prof. Dr. Bülent Karasözen	
Prof. Dr. Semra Kocabıyık	
Prof. Dr. Gerhard Wilhelm Weber	
Assoc. Prof. Dr. Tanıl Ergenç	
Assist. Prof. Dr. Hakan Öktem	

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Didem Akçay

Signature:

ABSTRACT

INFERENCE OF SWITCHING NETWORKS BY USING A PIECEWISE LINEAR FORMULATION

Didem Akçay
M.Sc., Department of Scientific Computing
Supervisor: Assist. Prof. Dr. Hakan Öktem
Co-supervisor: Prof. Dr. Semra Kocabıyık

December 2005, 87 pages

Inference of regulatory networks has received attention of researchers from many fields. The challenge offered by this *problem* is its being a typical modeling problem under insufficient information about the process. Hence, we need to derive the apriori unavailable information from the empirical observations. Modeling by inference consists of selecting or defining the most appropriate model structure and inferring the parameters. An appropriate model structure should have the following properties. The model parameters should be inferable. Given the observation and the model class, all parameters used in the model should have a unique solution (restriction of the solution space). The forward model should be accurately computable (restriction of the solution space). The model should be capable of exhibiting the essential qualitative features of the system (limit of the restriction). The model should be relevant with the process (limit of the restriction). A piecewise linear formulation, described by a switching state transition matrix and a switching state transition vector with a Boolean function indicating the switching conditions is proposed for the inference of gene regulatory networks. This thesis mainly concerns using a formulation of switching networks obeying all the above mentioned require-

ments and developing an inference algorithm for estimating the parameters of the formulation. The methodologies used or developed during this study are applicable to various fields of science and engineering.

Keywords: Inference, inferential modeling, statistical learning, piecewise linear systems, gene regulatory networks, hybrid systems.

ÖZ

PARÇALI DOĞRUSAL FORMÜLASYON KULLANILARAK DEĞİŞMELİ AĞLARIN ÇIKARIMI

Didem Akçay
Yüksek Lisans, Bilimsel Hesaplama Bölümü
Tez Yöneticisi: Yard. Doç. Dr. Hakan Öktem
Tez Yardımcısı: Prof. Dr. Semra Kocabıyık

Aralık 2005, 87 sayfa

Düzenleyici ağların çıkarımı birçok alandan araştırmacının dikkatini çekmektedir. Çıkarım problemlerinin önemi, modellenmesi düşünülen süreç hakkında yeterli bilginin olmadığı problemlerle ilgili olmasından kaynaklanmaktadır. Bu durumlarda ihtiyaç duyulan bilginin deneysel verilerden elde edilmesi gerekmektedir. Model çıkarımı; en uygun model sınıfının seçimi veya tanımlanması, ve model parametrelerin belirlenmesini içerir. Uygun bir model şu özellikleri taşımalıdır: Model parametreleri çıkarılabilir olmalıdır. Verili gözlem ve model sınıfı kullanılarak elde edilen parametrelerin tek bir çözümü olmalıdır (Çözüm uzayının sınırlanması). İleri adımlı model doğru şekilde hesaplanabilmelidir (Çözüm uzayının sınırlanması). Model, ilgilenilen sistemin gereken önemli özelliklerini ifade edebilmelidir (Sınırlamaya limit konması). Süreç ve model birbirine uyumlu olmalıdır (Sınırlamaya limit konması). Gen düzenleyici ağların çıkarımı için önerilen parçalı doğrusal formülasyon, değişen durum geçiş matrisi ve değişen durum geçiş vektörü ile kullanılan geçiş koşullarının gerçekleşmesinden sorumlu Boole fonksiyon ile tanımlanmaktadır. Bu tez temel olarak yukarıda belirtilen koşulları sağlayan değişmeli ağların formülas-

yonunu kullanmayı ve formülasyondaki parametrelerin tahmini için çıkarım algoritması geliştirmeyi amaçlamaktadır. Bu çalışma sırasında kullanılan yada geliştirilen metodların çeşitli mühendislik ve temel bilimler alanlarında uygulanabilirliği vardır.

Anahtar Kelimeler: Çıkarım, model çıkarımı, istatistiksel öğrenme, parçalı doğrusal sistemler, gen düzenleyici ağlar, hibrid sistemler.

To my family & Semih

ACKNOWLEDGMENT

I would like to thank my supervisor Assist. Prof. Dr. Hakan Öktem for giving me the opportunity to take a part in the project of *Modeling multistationary processes by using hybrid system formulation: A study with priority on functional genomics.* I am grateful for his guidance and supervision throughout this study.

I am thankful to my co-supervisor Prof. Dr. Semra Kocabıyık for her kindness and suggestions.

Special thanks to Prof. Dr. Gerhard Wilhelm Weber with whom I can always find a reason to smile. Thank you very much for your great friendship, understanding and your valuable guidance.

I am grateful to Prof. Dr. Bülent Karasözen for broadening my perspective both scientifically and socially.

I would also like to express my gratitude to Ömür Uğur and Tanıl Ergenç for their valuable ideas and suggestions.

Thanks to IAM (Institute of Applied Mathematics) members for making me feel myself in comfort during all my study in the institute.

This work is supported by TUBİTAK (The Scientific and Technical Research Council of Turkey) project no 104T133, *Modeling multistationary processes by using hybrid system formulation: A study with priority on functional genomics.*

Although I know that it is not completely possible to express my gratitude, I would like to thank to a few special people

... Özgür Hakanoğlu for his constructive suggestions and comments throughout this research.

... Selime Gürol for sharing hardest times of this study. You are more than a friend in my life. Your unstinting support and encouragement is very valuable in this study.

... Tunç Fındık to whom I owe so much in my way of thinking. It is a pleasure to have a climbing partner like you. I promise, I will never give up.

... My parents for their continuous and everlasting supports more than ever to keep my morality high enough during this challenging period. You are the best friends whom I can ever have in my life.

... Semih Perdahcıoğlu, to whom I owe my motivation. It is my precious gift to be with you, you are always very generous with your love, understanding and support as well as with your valuable being.

... thank all of you for being in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Modeling of Dynamical Systems

Physical systems can be observed, studied and measured using experimental devices. After such an analysis over the system of interest, the description of the behaviour of the observed system and its interpretation is delivered to mathematical models [6].

Nowadays, mathematical modeling is an indispensable tool for applied sciences. New and continuing developments in the applied sciences such as biology, genomics, ecology and climatology stimulated an interest in understanding and modeling the regulatory mechanisms underlying these dynamical systems [44]. Here, by the means of a dynamical system, the study of the long term behavior of evolving systems is emphasized [7]. The proper modeling approach depends on the prior information about the system, the behavior that is desired to model and so on.

First principles models attempt to describe the dynamic behaviour of the system from a detailed understanding of the components making up the system and their interconnections. This modeling approach leads to large systems of ordinary or partial differential equations, integral equations, integro differential equations, functional differential equations and differential algebraic equations. These equations are evolving in time continuously like the system they are intended to model so if sufficient knowledge about the system is available they lead to exact models. The

primary disadvantage of these equations is their immense complexity. Also the lack of information on a single interaction within the system, might result with totally incorrect predictions [47].

In some situations, sufficient information to build a first principles model is not available. This provides a strong motivation to consider alternative models. *Inferential modeling* deals with estimating the underlying dynamics from the empirical observations. In this case, the modeling approach consists of first selecting the most appropriate model class and, then, inferring the model parameters from the empirical data [44]. An appropriate model class should have the following properties:

- The model parameters should be uniquely inferable.
- The model should be capable of exhibiting the essential qualitative features of the interested system.
- The model should be relevant with the process.

Modeling attempts of natural phenomena lead to the development of mathematical model structures which in turn improve available modeling capabilities. It is crucial to realize that each problem needs its own considerations. This thesis mainly concerns with the inferential modeling of gene regulatory networks from gene expression data sets.

1.2 Problem Statement

Considerable evolution in microarray technologies allow simultaneous measuring of the transcription levels of thousands of genes evolving over time, reacting to different environmental or medical conditions [33].

While simultaneous measurements of thousands of gene expressions provides an access to the state of cells, integration of computational tools for data mining, visualization and statistical learning are essential to interpret the data. The elaboration of data mining and analytical tools are needed to show a causal relationship of the expression data, to group the data in a meaningful way and to perform statistical analysis in order to investigate its consequences [28].

Most of the publicly available gene data contains limited number of time point measurements with respect to a large number of genes. This brings requirement for additional simplifications.

1.3 Purpose and Scope

Gene regulation mechanism is a nonlinear multistationary system. It has more than one possible stationary states and sometimes a switching from one stationary state to another might occur [53, 61].

The main concept of this study is to use a piecewise linear formulation, described by a switching state transition matrix and a switching state transition vector with a Boolean function indicating the switching conditions, for the modeling of gene regulatory networks. Additionally, the aim is developing an inference algorithm for the estimation of model parameters and system states.

The second chapter of this thesis provides a basic introduction of gene regulation, brief definition of its units and the available data that come up with DNA microarray technology.

The third chapter describes some popular modeling approaches that are proposed to model gene regulatory networks in the literature.

The fourth chapter deals with the mathematical modeling of gene networks by using

piecewise linear equations. A step by step approach to piecewise linear equation models are given. Additionally the motivations for the selection of this model class and the explanation of this model is provided in that chapter.

The fifth chapter includes the inference algorithm and the procedures that are used to estimate the switching times and model parameters of each state.

The sixth chapter discuss the data requirement of the inference algorithm and the analysis of a 6 variables network.

Finally, a conclusion of the thesis by further discussions and outlooks on future perspectives is given in the seventh chapter.

CHAPTER 2

BACKGROUND

Genetic regulation is a very complex and complicated process when it is considered in molecular level [43, 66]. This chapter provides a basic introduction of genetic regulation occurred on transcriptional level, brief definition of its units that play a central role in this process and available data that come up with developing technology.

2.1 DNA, Genes and Proteins

2.1.1 DNA

A DNA molecule is a huge double helix made up of chains of chemical building blocks called nucleotides. Each nucleotide consists of a phosphate group, a deoxyribose sugar molecule and one of four different nitrogeneous bases referred by their initial letters: *Guanine* (G), *Cytosine* (C), *Adenine* (A), or *Thymine* (T). Genetic information is encoded in DNA by the sequence of these nucleotides (see Figure 2.1). It was proposed by Watson and Crick in 1953 that the two nucleotide chains are held together by hydrogen bonds that is formed between the nitrogeneous bases. Guanine hydrogen bonds with cytosine and adenine hydrogen bonds with thymine [39].

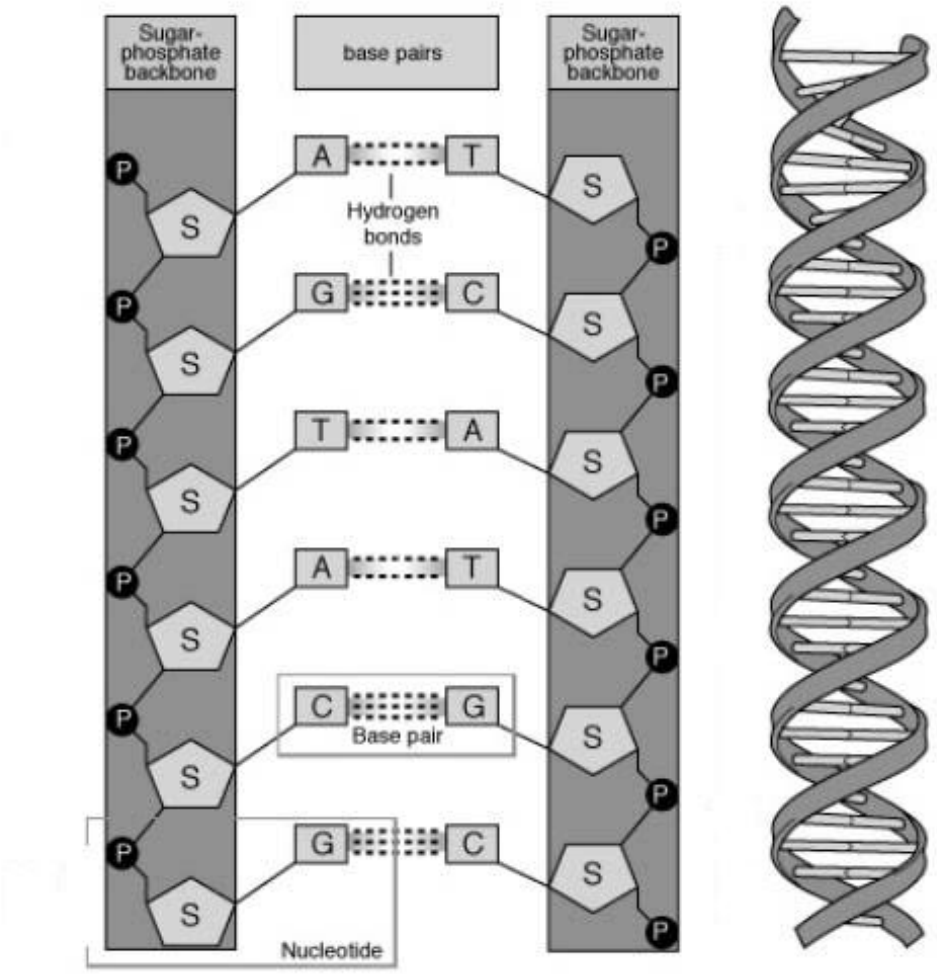

Figure 2.1: DNA structure [52].

2.1.2 Genes

A gene is a specific section of DNA which contains information that is used somehow to construct the organism [49]. There are two general types of genes. A *structural gene* contains information that specifies the primary structure of a protein. A *regulatory gene* is a section of DNA whose function is helping to control the access to the information in structural genes (see Figure 2.2) and it is usually near to a particular structural gene. In most cases there is a single gene that codes for each

protein, in other words each protein has its own gene [49].

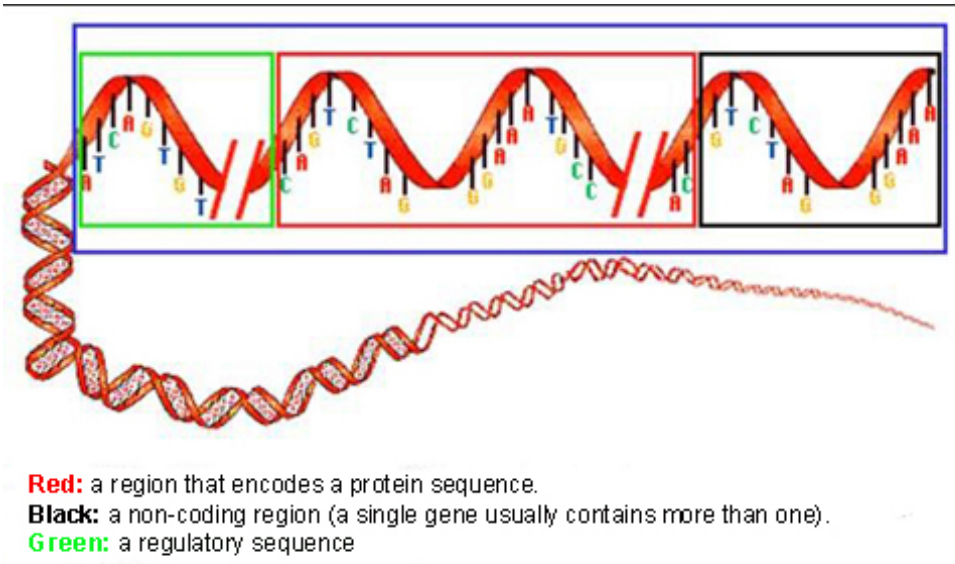

Figure 2.2: Gene structure [52].

2.1.3 Proteins

Proteins are the essential structural and metabolic components of the cell [57]. Almost every function in the body is controlled or promoted by some type of protein. There are many kinds of proteins serving different functions, such as hormone receptors, enzymes and transcription factors [49].

2.2 Gene Regulation is a Dynamical System

Each multicellular organism carries an identical set of genes. Every cell type of multicellular organisms exhibits a distinct behavior because the genes are expressed differently in different cell types. Thus, for understanding what makes a specific cell

type behave the way it does, we need to investigate how genes are regulated in that cell.

The genetic information is stored in DNA. Although DNA is very stable storage molecule, it is also rather passive molecule. For the information stored within the DNA to be utilized it must be transformed into a molecular form with a more active role. This is what is meant by *expressing* a gene. The first step in gene expression is to produce mRNA molecules by a process of *transcription*. In higher organisms, a structural gene's information specifying the primary structure of a particular protein must go to the place where proteins are synthesized. The molecule which transmits the genetic information to the place where proteins are synthesized is the *messenger RNA* (mRNA). Molecules of mRNA are synthesized by using the gene as a guide. The information contained in the mRNA is used to guide the synthesis of a protein which is a much more versatile type of molecule and this process is named as *translation*. The overall working mechanism until the end of the synthesis of a protein is referred to as *Central Dogma of Molecular Biology* [43].

The gene expression is regulated by many mechanisms at its different stages. These include mechanisms for controlling transcription initiation, RNA splicing, mRNA transport, translation initiation, post-translational modifications, and degradation of mRNA/protein.

One of the main junctions at which regulation occurs is mRNA transcription. In order for a gene to be transcribed into mRNA, it is often necessary for a specific protein to bind to regulatory regions along the DNA. Other proteins may also bind to the DNA if they have an effect on the rate of transcription, this proteins are known as *transcription factors* [56]. Since the transcription factors are the products of transcription and translation, this completes a feedback loop that allows genes to control each others' expression levels. In other words, this cycle defines a dynamic system involving highly nonlinear feedback mechanisms (see Figure 2.3).

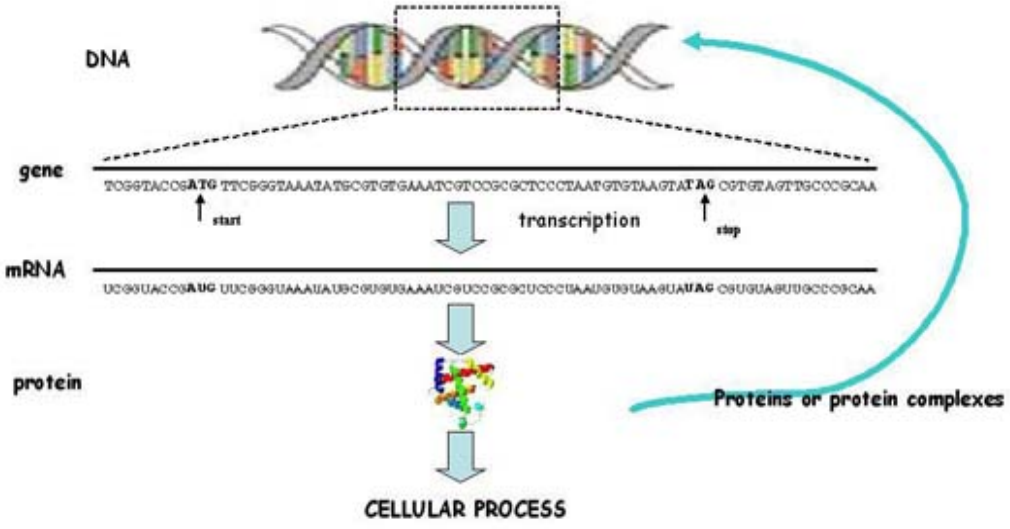

Figure 2.3: Gene regulation mechanism.

2.3 Gene Regulatory Networks

Genes regulate each other's activity by coding for transcription factors, which may enhance or repress the expression of other genes by binding (sometimes in form of complexes) at particular sites. Though a particular gene directly regulates just a small set of other genes, those genes regulate other genes in turn. So a gene will indirectly influence the activity of many genes downstream. Conversely, a particular gene is indirectly influenced by many genes upstream. A gene may directly or indirectly contribute to regulating itself. The result is a regulatory network, a complex feedback web of genes turning each other on and off [68].

2.4 Multistationarity of a Cell & Positive Feedback Circuits

Each cell of an organism has exactly the same DNA, yet there are many distinct kinds of cells in an organism. Also it is known that not all of the DNA information is used in each cell, and not all the information is used all of the time.

Max Delbruck suggested that identical DNA does not imply identical patterns of gene expression. Cell differentiation in organisms might be interrelated with the existence of distinct states of expression in the genetic regulatory networks of the cell [12, 61]. Each type of cell would then correspond to distinct states in the dynamics of the network of interacting genes and proteins. The process of differentiation corresponds to each cell, evolving to the state of its class. On the other hand, transition of the cell from one such state to another represents temporal changes in the metabolic state of a cell.

A gene, when expressed, leads to the production of proteins that can act to either activate or repress the activation of other genes. Within a network of such interacting genes and proteins, feedback mechanisms play a central role in controlling the dynamics [10, 53, 62]. For example, many transcription factors function in an autoregulatory capacity to control the expression of their own genes in either positive or a negative feedback loop [66].

It is long before recognized that negative circuits are necessary for stabilizing gene expression. For instance, the protein product of a gene can repress the synthesis of that gene. In this situation, when the concentration of the protein increases, it will decisively turn off its own synthesis [60, 61].

When it is analyzed from the window of positive circuits, the protein product of a gene exerts a positive action on its own synthesis. In the absence of that protein the gene is off and remains off. In the presence of that protein, the gene will be on

and since it is on, more product will be synthesized and the gene will remain on [60]. Actually, there are other factors which always come into play and positive feedback tends to be associated with the dynamics that switch the cell from one stable condition to another. In short, positive circuits are essential for multistationarity [37].

The answer to the question "Is it possible to have multiple stationary states within a genetic regulatory network without positive feedback circuits?" was first studied by Rene Thomas who suggested that multiple stationary states can exist in the presence of at least one positive feedback loop in 1984 [53]. His observation is proved by various researchers [25, 61] over the past years. At the moment, most biologists seem to accept that positive feedback is a regular component of gene regulatory networks, and strongly associated with cellular multistability [60, 61].

2.5 DNA Microarray Experiments

2.5.1 Functional Genomics

The purpose of functional genomics is to place all of the genes in the genome within a functional framework, both in the most basic terms of what the protein encoded by each gene does and also in the broadest sense of the role of each gene in the overall functioning of the cell and organism [33]. It is widely believed that thousands of genes and their products (mRNA and proteins) in a given organism function in a complicated way which is still preserved to be one of the mystery of life.

Technologies for simultaneously analyzing the expression levels (i.e., the amount of mRNA produced from a gene) of large amounts of genes provides the opportunity to study the activity of whole genomes, rather than the activities of single or a few genes. The technology gives the chance to look for groups of genes involved in a particular biological process or in a specific disease by identifying genes whose

expression levels change under certain circumstances [52].

2.5.2 DNA Microarray Technology

DNA microarrays have been developed as a method for rapidly analyzing the expression of all genes within a genome. DNA microarrays measure the expression of a gene in a cell by measuring the amount of mRNA present for that gene. A DNA microarray experiment starts with microarray construction in which the nucleotide sequences for a few thousand genes are printed on a glass slide. mRNA samples which are being transcription products, are then collected from a population of cells subjected to various experimental conditions. These samples are converted to cDNA and are labeled with one of two different fluorescent dyes (green or red) in the process. A single experiment consists of hybridizing the microarray with two differently labeled cDNA samples collected at different times. Generally, one of the samples is from the reference or background state of the cell, while the other sample represents a special condition set up by the experimenter. The level of expression of a particular gene is roughly proportional to the amount of cDNA that hybridizes with the DNA fixed to the slide. The relative levels of gene expression for any pair of conditions can be measured by using laser scanning flourescence detection technology [14, 39, 52]. The result, from an experiment with n DNA samples on a single chip, is a series of n expression-level ratios. Typically, the numerator of each ratio is the expression level of the gene in the condition of interest to the experimenter, while the denominator is the expression level of the gene in the reference state of the cell.

The data from a series of m such experiments may be represented as a gene expression matrix, in which each of the n rows consists of an m-element expression vector for a single gene [28, 65].

CHAPTER 3

MODELS OF GENE NETWORKS IN LITERATURE

3.1 Models of Gene Networks

Increasing volumes of data coming with DNA microarray technology directed researchers to analyze and explain the information underlying the available data. Various types of gene regulation network models have been proposed for trying to reveal the behavior of complex mechanisms included in gene regulation.

The models of gene regulation can vary from Boolean networks to biochemical interaction models with stochastic kinetics [4]. Former models are more tractable in mathematical sense and also it simplifies the analysis of systems with thousand of genes. On the other hand, the latter model supports the biochemical reality better, but the complexity of the model make it applicable to very small systems. As it is also exemplified in [13], a detailed biochemical model of *lysis-lsogeny switch in Lambda phage* [4] which has five genes with 67 parameters are decided at the end of nearly 50 years work and supercomputers are required for its stochastic simulation.

Biochemical modeling is necessary for understanding the molecular interactions underlying the regulatory mechanisms. Since a detailed molecular model is very complex for a network of 5 genes, it is very doubtful to construct it for a network of thousand genes.

As a result, researchers tend to more general methods for the interpretation of the gene expression data.

Comprehensive reviews [11, 21, 56] can be found in the literature discussing various methods that have been employed in mathematical biology and bioinformatics to describe genetic regulatory systems. In this chapter, three different popular modeling frameworks are introduced, namely Boolean networks, differential equation models and piecewise linear equation models.

3.2 Boolean Networks

A variable x that can take only two values is called *Boolean*. The values are usually denoted by 1 or 0, and they correspond to the logical values **true** and **false** respectively. The logic operators **and**, **or** and **not** are defined to correspond to the intuitive notion of truthfulness and composition of those operators. For example, x_1 **or** x_2 = **true** if and only if one of the terms x_1 **or** x_2 is true.

A *Boolean function* is a function of Boolean variables connected by logic operators. For example,

$$f(x_1, x_2, x_3) = x_1 \text{ and } (\text{not}(x_2 \text{ or } x_3))$$

is a Boolean function of three variables.

A *Boolean network* is a directed graph $G(X, E)$, where the nodes $x_i \in X$, are Boolean variables. To each node, x_i is associated by a Boolean function,

$$f_i(x_{i1}, x_{i2}, ..., x_{il}) \qquad (l \leq p,\, x_{ij} \in X),$$

where the arguments are all and only the parent nodes of x_i in G. At any given time, the values of all nodes represent the states of the network, given by the vector $s(t) = (x_1(t), x_2(t), ..., x_p(t))$.

For gene networks, the node variables correspond to gene expression levels present (1) and absent (0). The value of each node variable depend on prior activity of some other nodes and the states of all nodes are updated at the same time (synchronously) according to their Boolean functions

$$x_i(t+1) = f_i(x_{i1}(t), x_{i2}(t), ..., x_{il}(t))$$

corresponding to a state transition of the network from $s(t)$ to the new network state $s(t+1)$.

The series of state transitions are called *trajectories*; for example, $100 \rightarrow 000 \rightarrow 001 \rightarrow ... \rightarrow 001$ defines one trajectory of length 3 in the 3 variable network. The repeating parts of the trajectories are called *attractors* and can be one ore more states long. In this example, 001 is an attractor. All the states leading to the same attractor *are the basin of attraction*. Please refer to [68] for a detailed explanation of gene network modeling with basins of attraction. Wuensche developed a software named *Discrete Dynamics Lab (DDLab)* for the investigation of characteristics, parameters and measures of global dynamics, dynamics along particular trajectories, their size and distribution and topology, stability and adaptability to perturbations of network architecture and order chaos measures on network dynamics [68].

The goal in inference of Boolean networks is, the determination of the edges of the network and the Boolean functions of the nodes from gene expression data.

The amount of data needed to completely determine a unique network is known as the *data requirement problem* in network inference. Required data points for the inference of a gene network of p genes depends on the complexity of the model. Constraining the number of regulatory inputs per gene can drop the amount of required data significantly. The required data for fully connected and reduced connected cases of a Boolean network model is seen in Table 3.1. Please refer to [15] for the details.

Table 3.1: Fully connected: each gene can receive regulatory inputs from all other genes. Connectivity k: at most k regulatory inputs per gene.

MODEL	**Required data**
Boolean, fully connected	2^p
Boolean, connectivity k	$2^k(k + \log(p))$

In [2], *Akutsu et al.* showed that if the number of input nodes to each node is bounded by a constant, $O(\log p)$ state transition pairs (from 2^p pairs) are necessary and sufficient for identifying the Boolean network of p nodes correctly with high probability.

The dynamical properties of Boolean networks make them be attractive models of gene networks. Boolean networks can represent a complex behavior, and they can be characterized with stable and reproducible attractor states. The range of behaviors of the system is completely known and analyzable for smaller networks. However, these models are eventually limited by their definition. They are Boolean and synchronous. In reality, the levels of gene expression do not only have two states, they take continuous values. So, reducing the data values into two states may not be sufficient most of the time [21]. Additionally, biological networks are asynchronous, however, the updates of the network states are synchronous in Boolean networks.

For making Boolean network models biologically realistic, some extensions are made. In [47], a model class is introduced with Boolean interaction of binary state variables evolving in continuous time. With this model, *Öktem et al.* are able to describe epigenetic differences, adoption and learning mechanisms of evolutionary variation.

3.3 Differential Equation Models

Differential equations (DE) are the most widespread formalism for the quantitative modeling of complex systems. DE's are continuous and deterministic (non-stochastic) modeling formalism.

DE models of gene networks are based on rate equations. Rate equations determine the rate of change of a gene expression from the function of the expressions of the other genes.

If a gene network consists of p genes, the DE model will consist of p equations, one for each gene. The general form of the equations for the i^{th} gene is as follows:

$$\frac{d}{dt}E_i(t) = f_i(E_1(t), E_2(t), ..., E_p(t)). \tag{3.3.1}$$

The right-hand side identifies the rate of change in time. The function f_i describes how each of the p genes directly effects the transcription rate of the i^{th} gene. If f_i is positive, the change in the expression level of gene$_i$ is positive, if f_i is negative, the change in the expression level of gene$_i$ is negative, and if it is zero there is no change in the expression level. Hence, in some states of the network other genes are acting to switch on the i^{th} gene, and in other states they are switching it off. If $f_i(E_1(t), E_2(t), ..., E_p(t)) = 0$ for all $i = 1, 2, ..., p$, then the system is in a fixed point stationary state.

Additional terms like degradation of mRNA with time or other processes which can affect the activity of a gene can be included to the right-hand side of the equation (3.3.1). *D'haesseler et al.* [16] add an extra term to the right-hand side of the model indicating the influence of *kainate* (a glutamatergic against which causes seizures, localized cell death and severely disrupts the normal gene expression patterns) and a *constant bias term* to model the activation level of the gene in the absence on any other regulatory inputs.

In real biological systems, to utilize the measurements of transcription, translation and diffusion to the place of action of a protein an additional time delay can be introduced. This required time can be represented by discrete time delays as

$$\frac{d}{dt}E_i(t) = f_i(E_1(t-\tau_{i1}), E_2(t-\tau_{i2}), ..., E_p(t-\tau_{ip})) \qquad (1 \leq i \leq p),$$

where $\tau_{i1}, \tau_{i2}, ..., \tau_{ip} > 0$ denote discrete time delays [11].

Identifying a gene network from gene expression data means estimating any existing parameters in the functions $f_i(.)$. The simplest case is that the $f_i(.)$ are linear

$$\frac{d}{dt}E_i(t) = b_i + m_{i1}E_1(t) + m_{i2}E_2(t) + ... + m_{ip}E_p(t), \tag{3.3.2}$$

where the matrix notation of the whole system is

$$\frac{d}{dt}E(t) = ME(t) + b.$$

In equation (3.3.2), the parameter b_i represents the background level of transcription of the i^{th} gene [54]. Here, m_{ij} describes the regulatory effect of gene$_j$ on gene$_i$ and corresponds to the strength of this effect. If the j^{th} gene does not directly effect the transcription rate of the i^{th} gene then $m_{ij} = 0$. If the j^{th} gene does not effect the i^{th} gene directly, the E_j will not appear as an argument in $f_i(.)$, and $f_i(.)$ will only be a function of a small subset of $E_1, E_2, ..., E_p$.

In the literature, various differential equation modeling approaches to the dynamic nature of gene expression time series exists.

Chen et al. [9] proposed a system of differential equations $\frac{d}{dt}E(t) = ME(t)$, where E is a vector of mRNA's and protein concentrations at time t and M is a constant matrix representing regulatory interactions for both mRNA's and proteins.

Due to the fact that from the DNA microarray measurements only the mRNA concentrations of the corresponding genes are available, whereas the protein concentra-

tions are not known, many researchers build their models on the basis of mRNA data. The simplicity of linear differential equation models make them preferable for researchers. *Hoon et al.* [32] proposed a system of linear differential equations in the form of

$$\frac{d}{dt}E(t) = ME(t)$$

and in [31] applied this model on the mRNA data of *Bacillus subtilis* where a coefficient m_{ij} of the coefficient matrix M represents the effect of gene j on gene i. The number of nonzero entries in M was determined from the data by using *Akaikes Information Criterion* [28].

In [24], the approach of *Hoon et al.* and *Chen et al.* is extended by letting the matrix M depend on E where, a system of differential equations is defined as

$$\frac{d}{dt}E(t) = M(E(t))E(t).$$

The algorithm presented by *Gebert, Latsch, Pickl, Weber and Wunschiers* [22] detects the parametrical regions of stability and unstability of this time continuous system.

Sakamoto and Iba [50] choose an arbitrary form in the right-hand side of the system of differential equations to allow flexibility of the model. They consider the following form

$$\frac{d}{dt}E_i(t) = f_i(E_1(t), E_2(t), ..., E_p(t))$$

and infer the right-hand sides by using *genetic programming*.

Yılmaz [70] modeled gene expression patterns by ordinary differential equations as in the approach of *Sakamoto and Iba*, by letting the right-hand side $f(E)$ component wisely consists of a sum of non-linear functions. In [70], special attention is paid to the case of quadratic model functions. They behave superior to linear ones with respect to both accuracy of data fitting and quality of future state prediction.

Taştan [58] developed the models of [24, 70] with an affine term $C(E)$ where the system takes the form of

$$\frac{d}{dt}E(t) = M(E(t))E(t) + C(E(t)).$$

For the linear differential equation models, if the matrix M is dense, $p+1$ arrays of p genes are needed to solve the system. If the average connectivity per node is a fixed constant k as in the case of real gene networks then $k \log p$ experiments are needed [15, 59, 69].

When the gene expression measurements have been performed around a steady state or on a slow changing system, linear models yield good predictions. Otherwise the rates of change of the expression levels of genes cannot be estimated well [21].

Since most of the gene regulatory interactions are nonlinear, one way to approach the systems steady states is to identify the system with the collaboration of Boolean networks and differential equations which will also establish our modeling approach to gene regulatory networks.

3.4 Piecewise Linear Equation Models

Linear models of gene networks yield good predictions in the close vicinity of steady states. However, they are not flexible enough to model nonlinear behavior of gene networks.

Boolean networks allow large regulatory networks to be analyzed by making strong simplifying assumptions on the structure and dynamics of a genetic regulatory system. They assume that the transitions between the activation states of the genes occur synchronously, while it is asynchronous in reality.

The use of piecewise linear equation models combines the linear equation models

and the Boolean networks to profit from both methods in terms of computational effectiveness.

Gene expression measurements are real-valued and change continuously in time; therefore, the rules that specify the time derivatives of expression levels can take logical forms. Piecewise linear equation models can represent many of the complex, nonlinear regulatory phenomena observed in real gene networks.

The simplest piecewise linear system can be represented as follows

$$\frac{d}{dt}E(t) = M_{s(t)}E(t) + b_{s(t)} \tag{3.4.1}$$

$$s_i(t) = \begin{cases} 1 & \text{if } E_i(t) \geq thr_i \\ 0 & \text{if } E_i(t) < thr_i \end{cases},$$

where M is a matrix valued function and b is a vector valued function which can take varying values depending on the state vector s. Here, thr_i is the corresponding threshold of the i^{th} gene. The thresholds of the variables divide the state space into 2^p subspaces (p is the number of variables) where each subspace is governed by a piecewise linear differential equation.

Edwards et al. [18, 19] proposed a system of piecewise linear equation model where M is taken as a constant diagonal matrix with diagonal entries of -1. Furthermore, b has the form of $b_i = f_i(s(t))$, where $s(t)$ is a vector containing the logical states of the regulators of gene$_i$ for the state of $s(t)$. The values of $f_i(.)$ define focal points determined by the current state of the system. Threshold values of the variables are taken as 0 and focal points are defined as **-1** or **1** for the corresponding variables. Hence, the dynamics of the system is specified by b which allows most of the non-linear behavior of gene networks to be represented in that way.

Perkins et al. [48] applied the same approach as *Edwards et al.* did. However, in [48], threshold values of each variable are taken as 1/2 and focal points are defined to be 0 or 1. The detailed analysis of piecewise linear systems can be found in [17].

More general forms of equation (3.4.1) exists in the literature [23, 26, 35, 36]. In these models, more than two logical levels per variables are introduced and these variables are separated by arbitrary real thresholds where each threshold is defined for production and decay rates. In [34], the application of the method to model the *sporulation network* and simulation of the *response of the cell nutrient derivation* can be found.

In our modeling approach, we restrict our attention to the simplest form of the models given in equation (3.4.1), where the transition matrix M and the transition vector b are the parameters of the corresponding state. Also the threshold values of the variables are unknown and extracted from the available data. The details of our model will be given in the proceeding chapter.

For the piecewise linear differential equation models when identifying b is assumed to be a statistically independent process for each gene, the expected number of switches before the whole vector b for the system states can be identified is pH_p2^k, where k represents the number of regulators, p is the number of genes and $H_p = 1+\frac{1}{2}+...+\frac{1}{p}$ is the p^{th} harmonic number [48].

A network of p genes, each regulated by k genes can be connected in

$$\left(\begin{array}{c} p \\ k \end{array} \right)^p \simeq p^{pk} \tag{3.4.1}$$

different ways [48]. When the network in consideration is larger, there is an excessive increase in the total number of different networks that capture network structure. As it is discussed in [18], for 3 dimensions there are 112 different networks, for 4 dimensions there are 10^7 ones, and in 5 dimensions 3×10^{20} different networks are found.

CHAPTER 4

MATHEMATICAL MODELING

4.1 The Regulatory Network Inference Problem

Developing technology for monitoring expression profiles, genomic sequences and others gives the opportunity to measure the activity of thousands of genes simultaneously evolving in time or respond to different environmental, pharmaceutical or genetic conditions. Such an access to the states of cells offer more accurate inference of the network of interactions that regulates gene expression [48].

In reality, gene expressions are real-valued and are evolving continuously in time. The time evolution of p state variables (i.e., genes) can be represented by a vector valued function

$$E(t) = (E_1(t), E_2(t), ..., E_p(t))^T.$$

However, a restricted number of bits in computers for storing the information allows us to store only a finite number of samples of continuously evolving system. DNA microarray measurements are observed from sampling of gene expression levels in time. For this study, a uniform (equidistant) sampling is assumed.

Gene expression time series might be represented as

$$E(n) := E(nt_s) \qquad (n \in \mathbb{N}),$$

where $E(n)$ is the expression level vector for the n^{th} time sample with sampling

periods of length t_s. To be more precise, the symbol $E(n)$ is used as an approximative time-discrete description of the value $E(nt_s)$ which the time-continuous system takes at the sample time nt_s.

The regulatory network inference problem deals with the inference of the dynamics of the corresponding system from the available data. Assume that the correct real function f^* is underlying the data and our goal is to find a simple parametric function f_p that accurately approximates f^*. This problem can be summarized as follows:

- $\sum_{n=1}^{m}(E(n+1) - f_p(E(n)))^2$ is minimum,
- f_p is as simple as reasonably possible.

As an error function, *least square error (LSE)* is selected due to the fact that its solutions are linear functions which are analytically solvable.

In this thesis, the relation between present and next states of the gene expressions are preferred to be explained by difference equations $E(n+1) = f(E(n))$, rather than differential equations.

4.2 Appropriate Model Class Selection for Gene Regulatory Dynamics

Missing information, noisy measurements, apparent complexity of gene regulation mechanism make the inference problem difficult [56]. Because of these difficulties the main questions that arise are:

- Is it possible to infer gene regulatory dynamics from expression time series data?

- If so, what inference algorithms are preferable?
- How much data are required to infer the system dynamics?

Theoretical answers to these questions depend on how one models gene regulation and expression dynamics [48].

It is critical that the selected model should be computable. Additionally, if a poor model structure is chosen, the outcoming numerical results will not have any significant use.

Qualitative behavior matching is a useful criterion for model structure selection. This approach is based on the selection of the most appropriate model class that is capable of exhibiting the qualitative behavior of primary interest [44].

In addition, for the interpretation of the numerical results, the model variables need to be relevant for the investigated process [3, 44].

Inference of gene regulatory dynamics is a typical *inverse problem* [5]. We have the temporal evolution of expression levels of genes (mRNA concentrations $E(n)$ for the n^{th} time sample) from DNA microarray measurements. We would like to infer the system dynamics by defining the most appropriate model function f that represents the relation between previous and next states of expression levels (see Figure 4.1).

Let us study some cases in order to find the appropriate model which defines the regulatory mechanisms underlying gene networks.

4.2.1 When f is Taken as Any Function

When f is taken as any function, adoption to a very wide range of dynamics is possible. Infinitely many different functions fitting to an available data set can be found thus selecting the correct particular model is not possible.

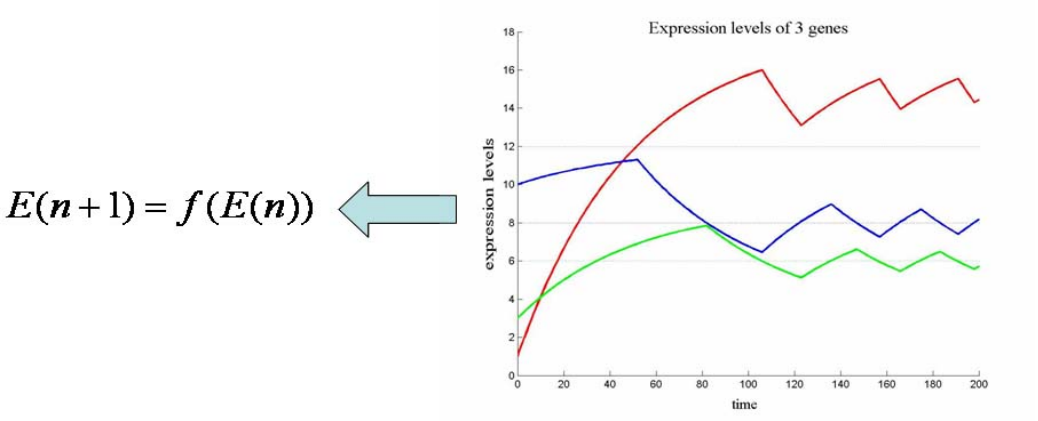

Figure 4.1: Inference of gene regulatory dynamics is an inverse problem.

For finding the unique solution to the inverse problem, the selected model class should restrict the solution space.

4.2.2 When f is Taken as a Linear Function

When f is taken as a *linear* function, the primary function takes the following form:

$$E(n+1) = ME(n) \qquad (n \in \mathbb{N}).$$

In this case, model parameter M can be uniquely determined by using least squares method (the coefficient matrix of the linear system is supposed to be nonsingular-it will be clarified in Section 5.2); so a unique optimum solution can be found to the inverse problem [29]. The equation $E(n+1) = ME(n)$ can be extended without losing generality of linear system features as follows

$$E(n+1) = ME(n) + b \qquad (n \in \mathbb{N}),$$

where b is a vector shifting the origin of a linear system.

However, most of the regulatory interactions are *nonlinear* in structure. This means that the rate of the controlled process is not a linear function of the concentration of the regulator.

Most biological regulatory interactions -genetic regulation is also included in this class- are sigmoid in shape [60]. Consider, for example, a gene whose expression depends on the presence of a positive regulator. When the concentration of the regulator increases, the rate of expression of the gene is first insignificant, then it sharply raises within a rather narrow range and then levels off (see Figure 4.2).

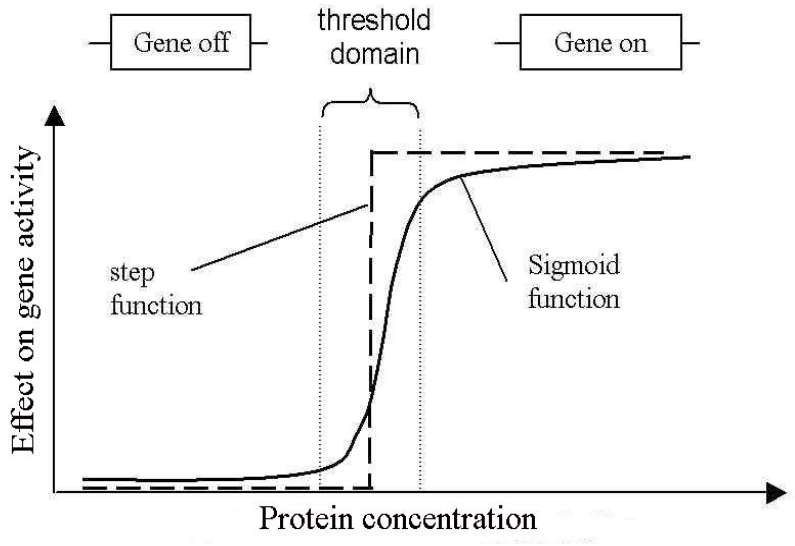

Figure 4.2: A non-linear sigmoid shape interaction of gene regulation.

As regulatory interactions are nonlinear in structure, the equations used for their description are nonlinear. This proves the fact that linear functions are not sufficient to hold the regulatory behavior.

4.2.3 When f is Taken as a Piecewise Linear Function

Nonlinear structure of regulatory interactions strongly implies to consider some idealizations for simplifying the numerical computations [60].

The most obvious one consists of a linear approximation. As it is also mentioned in Section 3.3 that this simplification is valid in the close vicinity of stationary steady states, but disastrous elsewhere.

Another idealization is to simplify the sigmoid shape function by logical descriptions. Logical descriptions use variables which can take only a limited number of values, most often only two: 0 and 1 (in this case, one speaks of a "Boolean," or binary, description; see Section 3.2). In biology, a regulator is often inefficient below a "threshold" range of concentration, and its effect rapidly levels up above this threshold. In the simplest cases, this description treats a gene as "ON" (1) if it is over its threshold, or "OFF" (0) if it is under its threshold [61].

In piecewise linear equation models, *linear approximation* and *logical description* idealizations are combined. In an approach based on a class of piecewise linear equation models, piecewise linear equations represents the dynamics of the system states and the use of the Boolean functions is motivated by the nonlinear, switch-like character of many of the interactions in gene expression.

4.3 Piecewise Linearity

Let the state space $U \subseteq \mathbb{R}^p$ of a dynamical system is formed by z disjoint subspaces such that

$$U = U_1 \cup U_2 \cup ... \cup U_z$$

and

$$U_i \cap U_j = \phi \qquad \text{if } i \neq j.$$

Consider three different initial states which are elements of the same subspace U_i. This means

$$y_0 \in U_i,\ y_1 \in U_i,\ y_2 \in U_i,$$

which satisfies the following equality:

$$y_2 - y_0 = r(y_1 - y_0) \qquad (r \in \mathbb{R}).$$

Let y_0, y_1, y_2 yield $y_0(t)$, $y_1(t)$, $y_2(t)$, respectively. This means: If the system started with the initial state $y(t_0) = y_0$ then the function representing its temporal evolution for $t > t_0$ is denoted by $y_0(t)$.

The system is piecewise linear in U_i if

$$y_2[t_0, t_i] - y_0[t_0, t_i] = r(y_1[t_0, t_i] - y_0[t_0, t_i]) \qquad (r \in \mathbb{R})$$

and $y_0(t), y_1(t), y_2(t) \in U_i$ for all $t_0 < t < t_i$ where $[t_0, t_i]$ stands for the time interval that is spent in subspace U_i. Herewith, the system itself is said to be *piecewise linear* if it is piecewise linear in all subspaces of its state space [45].

4.4 Switching Difference Equations

A piecewise linear system can be represented by *switching difference equations* as follows [45]:

$$y(n+1) = M_{s(n)} y(n) + b_{s(n)} \tag{4.4.1}$$

$$s_i(n) = \begin{cases} 1 & \text{if} \quad y_i(n) \geq thr_i \\ 0 & \text{if} \quad y_i(n) < thr_i \end{cases}$$

where $y(n)$, $y(n+1)$ represent the present and next sample values of the variables, $s(n)$ is a Boolean vector representing the state of the system at n^{th} time sample. Furthermore, $M. : \{0,1\}^p \rightarrow \mathbb{R}^{p \times p}$ is a matrix-valued function (state transition matrix), $b. : \{0,1\}^p \rightarrow \mathbb{R}^p$ is a vector-valued function (state transition vector) whose elements are determined by the state of the system. Finally, i denotes the i^{th} element

of the corresponding vector, and thr_i is the threshold of the corresponding variable.

Equation (4.4.1) defines a system with p variables, where only one threshold is defined for each variable. So, the state space of the system has 2^p subspaces (i.e., states) and within each subspace the governing piecewise linear difference equations might vary.

For any complex nonlinear dynamical system satisfying the conditions that

- the state space can be partitioned into a finite number of subspaces and the governing equations of each subspace can be analytically computable,
- in a finite time, the number of the switchings are finite,

can be represented by switching difference equations which is based on piecewise linear equations (describing the evolution of the systems continuous variables) and Boolean functions (representing evolution of the system states) [46].

4.5 Why Piecewise Linear Formulation for Gene Regulatory Dynamics

As it was mentioned in Section 4.2.2, gene regulatory interactions are high-dimensional and nonlinear in structure. Also we know that in biology it is common to imagine that genes are switched "on" or "off".

Genomic regulation can be treated as a multistationary system which may exhibit more than one possible stable state and can perform a transition from one stable state to another. The state of a gene can be represented by a Boolean vector. If the expression level of the concerned gene is over its threshold then it is *on*, otherwise it is *off*. Such a logical regulation over the system state makes it possible to partition system into tractable subspaces which provides to investigate the overall

system easily [1]. In Section 3.3, it was pointed out that in the close vicinity of the stationary steady states, linear equations are proper tools for representing the system evolution. A stationary steady state is based on the property that the mean, variance and the autocorrelation structure do not change over time, while time goes to infinity [3]. In the selected tractable subspaces, logical states of the genes and their production rates are constant. Hence, piecewise linear equations can be used to describe the concentrations of gene products, such as mRNA and proteins in that tractable subspaces. Such an approach provides a detailed understanding of the dynamical behavior exhibited by regulatory mechanisms.

As a conclusion: Piecewise linear models provide a coarse-grained description of genetic regulatory networks, well-adapted to state of the art measurement techniques in genomics [20, 35, 51].

4.6 Piecewise Linear Equation Model of Gene Regulatory Dynamics

In our model, $E(n) \in \mathbb{R}^p$, $E(n+1) \in \mathbb{R}^p$ represent the mRNA concentrations at n^{th}, $(n+1)^{th}$ time samples and $s(n) \in \{0,1\}^p$ is a Boolean vector representing the states of the genes (0 or 1) for the n^{th} sample.

The time evolution of a gene regulatory dynamics is represented as follows:

$$
\begin{aligned}
E(n+1) &= M_{s(n)}E(n) + b_{s(n)} \qquad (4.6.1)\\
s(n) &= F_B(Q(E_1(n-1), E_2(n-1), ..., E_p(n-1)))\\
Q_i(E(n)) &= \begin{cases} 1 & \text{if} \quad E_i(n) \geq thr_i \\ 0 & \text{if} \quad E_i(n) < thr_i \end{cases}
\end{aligned}
$$

where i is the i^{th} gene or the corresponding mRNA concentration.

Here, $M. : \{0,1\}^p \rightarrow \mathbb{R}^{p \times p}$ is a matrix-valued function (state transition matrix) and $b. : \{0,1\}^p \rightarrow \mathbb{R}^p$ is a vector-valued function (state transition vector) whose elements can take different values depending on the state of the system (i.e., depending on the state vector s). Furthermore, thr_i is the corresponding threshold for each gene. Finally, $Q : \mathbb{R}^p \rightarrow \{0,1\}^p$ is a quantized expression level vector which assigns 0 or 1 value to a gene in order to check if corresponding gene passed its threshold or not.

According to the study of Jacob and Monod [42], gene activation or inhibition is controlled by protein concentration. From DNA microarray measurements only mRNA concentrations are available. Therefore, in the formulation a single sampling time difference is used for representing the synthesis of proteins from the corresponding mRNA.

In the formulation $F_B : \{0,1\}^p \rightarrow \{0,1\}^p$ is a Boolean function of the quantized expression level vector $Q(E(n))$.

A time at which the state of any gene is changed is called a *switching time*. In the model, switchings are controlled by activation or inhibition of a gene. Activation or inhibition is controlled by whether the level of any transcription factor is crossing its threshold value. Between the switching times, the logical states of the genes and also the difference equations governing their concentrations are constant. Each concentration changes according to the formulation $E(n+1) = M_{s_m} E(n) + b_{s_m}$ with the state of s_m, where $s_m \in \{0,1\}^p$. The production rates can change at a switching time, which causes the dynamics of the system to be controlled by another transition matrix and vector according to the new state of the system.

CHAPTER 5

NETWORK INFERENCE

5.1 Inference of Gene Networks

As it is mentioned in Section 4.1, the regulatory network inference problem is based on determining the structure of the dynamics, threshold values of each gene, transition matrices and vectors that correspond to the states of the unknown system behaving according to equation (4.6.1) from gene expression data of DNA microarray measurements. The solution of this inference problem is the main aim of this thesis.

For the inference of the network structure, our approach is based on the following observation. While the concentration of the genes are evolving in time, if the change of the concentration of one of the genes is reversed, then it is deduced that the concentration of one of the other genes has passed its threshold in that instant. Therefore, it can be inferred that gene_j (passing its threshold) regulates gene_i [48].

So we can concluded that gene_j regulates gene_i if in one of the concentration time series

(i) there is a switching time at which j is the only gene for which there occurs a switching in its logical state;

(ii) at that switching time, the production rate of gene_i changes, i.e., when its expression level is rising, it starts to fall, or vice versa.

To estimate the model parameters and switching times of each state, *the least squares method* is used. For the detection of threshold values an approach based on mean and standard deviations is applied to the data vectors detected at switching times. Our approach will be clarified in the following sections.

5.2 Parameter Estimation

5.2.1 Least Squares Method

A multidimensional data corresponding to a multistate system comes up with gene regulatory interactions (see Section 4.5). Suppose a system with p variables and its gene expression level vectors $E(n)$, $n = 1, 2, ..., h$, is given by DNA microarray experiments and suppose that these vectors corresponds to one state of the system, i.e., $s(1) = s(2) = ... = s(h) = s_m$.

The problem of finding the equation system of the best approximation to the h experimental data in the least squares sense [8, 40, 64] requires that values of $M = [m_{ij}]_{\substack{i=1,2,...,p \\ j=1,2,...,p}} \in \mathbb{R}^{p\times p}$ and $b = [b_i]_{i=1,2,...,p}$ be found to minimize

$$[\widehat{M}, \widehat{b}] = \arg\min_{M,b} \sum_{n=1}^{h-1} (E(n+1) - (b_{s_m} + M_{s_m} E(n)))^2. \tag{5.2.1}$$

That is, to measure how well the model (equation (4.6.1)) fits to the data, the *least squares method* is used which minimizes the sum of the squares of the differences between the E-values on the approximating discrete trajectory and the given E-values. Hence, constants $M = [m_{ij}]_{\substack{i=1,2,...,p \\ j=1,2,...,p}} \in \mathbb{R}^{p\times p}$ and $b = [b_i]_{i=1,2,...,p} \in \mathbb{R}^p$ must

be found that minimize the least squares error [8]

$$Err_1 = \sum_{n=1}^{h-1}(E_{1,n+1} - \{b_1 + m_{11}E_{1,n} + m_{12}E_{2,n} + ... + m_{1p}E_{p,n}\})^2 \tag{5.2.2}$$
$$Err_2 = \sum_{n=1}^{h-1}(E_{2,n+1} - \{b_2 + m_{21}E_{1,n} + m_{22}E_{2,n} + ... + m_{2p}E_{p,n}\})^2$$
$$...$$
$$Err_p = \sum_{n=1}^{h-1}(E_{p,n+1} - \{b_p + m_{p1}E_{1,n} + m_{p2}E_{2,n} + ... + m_{pp}E_{p,n}\})^2$$

where $E_{i,n\ \substack{i=1,2,...,p \\ n=1,2,...,h}}$ represents the i^{th} variable of the n^{th} time sample.

As a minimization problem, a least squares problem can be treated using methods of multivariate calculus. The derivative of each row of equation (5.2.2) is calculated according to its parameters and then equated to zero [29]. For the first row of equation (5.2.2)

$$\frac{dErr_1}{db_1} = -2\sum_{n=1}^{h-1}(E_{1,n+1} - \{b_1 + m_{11}E_{1,n} + m_{12}E_{2,n} + ... + m_{1p}E_{p,n}\}) = 0 \tag{5.2.3}$$
$$\frac{dErr_1}{dm_{11}} = -2\sum_{n=1}^{h-1}(E_{1,n+1} - \{b_1 + m_{11}E_{1,n} + m_{12}E_{2,n} + ... + m_{1p}E_{p,n}\})E_{1,n} = 0$$
$$...$$
$$\frac{dErr_1}{dm_{1p}} = -2\sum_{n=1}^{h-1}(E_{1,n+1} - \{b_1 + m_{11}E_{1,n} + m_{12}E_{2,n} + ... + m_{1p}E_{p,n}\})E_{p,n} = 0$$

is obtained. Equation (5.2.3) can be written in the following form:

$$\begin{bmatrix} \sum_{n=1}^{h-1} 1 & \sum_{n=1}^{h-1} E_{1,n} & \dots & \sum_{n=1}^{h-1} E_{p,n} \\ \sum_{n=1}^{h-1} E_{1,n} & \sum_{n=1}^{h-1} (E_{1,n})^2 & \dots & \sum_{n=1}^{h-1} E_{1,n}E_{p,n} \\ \dots & \dots & \dots & \dots \\ \sum_{n=1}^{h-1} E_{p,n} & \sum_{n=1}^{h-1} E_{1,n}E_{p,n} & \dots & \sum_{n=1}^{h-1} (E_{p,n})^2 \end{bmatrix} \begin{bmatrix} b_1 \\ m_{11} \\ \dots \\ m_{1p} \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{h-1} E_{1,n+1} \\ \sum_{n=1}^{h-1} E_{1,n+1}E_{1,n} \\ \dots \\ \sum_{n=1}^{h-1} E_{1,n+1}E_{p,n} \end{bmatrix} \quad (5.2.4)$$

When the same procedure is applied to $Err_2, Err_3, ..., Err_p$, then the general form

$$\begin{bmatrix} \sum_{n=1}^{h-1} 1 & \sum_{n=1}^{h-1} E_{1,n} & \dots & \sum_{n=1}^{h-1} E_{p,n} \\ \sum_{n=1}^{h-1} E_{1,n} & \sum_{n=1}^{h-1} (E_{1,n})^2 & \dots & \sum_{n=1}^{h-1} E_{1,n}E_{p,n} \\ \dots & \dots & \dots & \dots \\ \sum_{n=1}^{h-1} E_{p,n} & \sum_{n=1}^{h-1} E_{1,n}E_{p,n} & \dots & \sum_{n=1}^{h-1} (E_{p,n})^2 \end{bmatrix} \begin{bmatrix} b_j \\ m_{j1} \\ \dots \\ m_{jp} \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{h-1} E_{j,n+1} \\ \sum_{n=1}^{h-1} E_{j,n+1}E_{1,n} \\ \dots \\ \sum_{n=1}^{h-1} E_{j,n+1}E_{p,n} \end{bmatrix} \quad (5.2.5)$$

is obtained where j represents the j^{th} variable.

In equation (5.2.5), the quadratic left-hand side matrix of type $(p+1)\times(p+1)$ may briefly be denoted as K such that

$$K := \begin{bmatrix} \sum_{n=1}^{h-1} 1 & \sum_{n=1}^{h-1} E_{1,n} & \dots & \sum_{n=1}^{h-1} E_{p,n} \\ \sum_{n=1}^{h-1} E_{1,n} & \sum_{n=1}^{h-1} (E_{1,n})^2 & \dots & \sum_{n=1}^{h-1} E_{1,n}E_{p,n} \\ \dots & \dots & \dots & \dots \\ \sum_{n=1}^{h-1} E_{p,n} & \sum_{n=1}^{h-1} E_{1,n}E_{p,n} & \dots & \sum_{n=1}^{h-1} (E_{p,n})^2 \end{bmatrix}_{(p+1)\times(p+1)},$$

and the right-hand side vectors can be collected in one matrix of type $(p+1)\times p$,

called F:

$$F := \begin{bmatrix} \sum_{n=1}^{h-1} E_{1,n+1} & \sum_{n=1}^{h-1} E_{2,n+1} & \dots & \sum_{n=1}^{h-1} E_{p,n+1} \\ \sum_{n=1}^{h-1} E_{1,n+1}E_{1,n} & \sum_{n=1}^{h-1} E_{2,n+1}E_{1,n} & \dots & \sum_{n=1}^{h-1} E_{p,n+1}E_{1,n} \\ \dots & \dots & \dots & \dots \\ \sum_{n=1}^{h-1} E_{1,n+1}E_{p,n} & \sum_{n=1}^{h-1} E_{2,n+1}E_{p,n} & \dots & \sum_{n=1}^{h-1} E_{p,n+1}E_{p,n} \end{bmatrix}_{(p+1)\times p}$$

where each column of F stand for the right-hand side of equation (5.2.5). In the same manner, each row of b and M can be collected in the columns of matrix u of type $(p+1) \times p$, where

$$u := \begin{bmatrix} b_1 & b_2 & \dots & b_p \\ m_{11} & m_{21} & \dots & m_{p1} \\ \dots & \dots & \dots & \dots \\ m_{1p} & m_{2p} & \dots & m_{pp} \end{bmatrix}_{(p+1)\times p} .$$

Then, the systems takes the form

$$Ku = F$$

where $K \in \mathbb{R}^{(p+1)\times(p+1)}$, $F \in \mathbb{R}^{(p+1)\times p}$ and $u \in \mathbb{R}^{(p+1)\times p}$. This form stands for the system of *normal equations* for the solution of least squares problem equation (5.2.1).

A sufficient condition that u is a minimum is that the Hessian matrix of second partial derivatives of equation (5.2.2) is positive definite which implies in turn that, K is positive definite [29].

A Matlab function *leastsquares.m* (Appendix B) is developed for dealing with this problem where Matlab's *backslash operator*, \, is used to solve the obtained system of normal equations, $Ku = F$. The description of the Matlab command "\" (backslash

operator) is given in Appendix A.

In references [18] and [48], the authors assumed that the production rate of any gene within a state depends only on its own concentration. This assumption decrease the number of independently adjustable parameters in the optimization problem (equation (5.2.1)) which is preferable when the number of observations are limited. In this case, at a particular state, the concentration levels of the other genes are assumed to have no effect on the production rate of the gene in consideration. This implies that the off-diagonal terms of the matrix M are set to zero.

In this case, equation (5.2.2) takes the form of

$$\begin{aligned} Err_1 &= \sum_{n=1}^{h-1}(E_{1,n+1} - \{b_1 + m_{11}E_{1,n}\})^2 \\ Err_2 &= \sum_{n=1}^{h-1}(E_{2,n+1} - \{b_2 + m_{22}E_{2,n}\})^2 \\ &... \\ Err_p &= \sum_{n=1}^{h-1}(E_{p,n+1} - \{b_p + m_{pp}E_{p,n}\})^2. \end{aligned}$$

When the same procedure (the derivative of $Err_{i_{i=1,2,...,p}}$ according to its parameters is calculated and then equated to zero) is followed as in the full matrix case, the normal equations

$$\begin{bmatrix} \sum_{n=1}^{h-1} 1 & \sum_{n=1}^{h-1} E_{j,n} \\ \sum_{n=1}^{h-1} E_{j,n} & \sum_{n=1}^{h-1} (E_{j,n})^2 \end{bmatrix} \begin{bmatrix} b_j \\ m_{jj} \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{h-1} E_{j,n+1} \\ \sum_{n=1}^{h-1} E_{j,n+1}E_{j,n} \end{bmatrix}$$

are obtained, where $j = 1, 2, ..., p$. A Matlab function *lstsqr.m* (Appendix E) is developed to deal with this problem, and its working procedure is similar to *least-squares.m*. *lstsqr.m* can run with multi-dimensional data; the parameters M and b are its outputs.

It is critical to realize that at least 3 data vectors are needed for the estimation of M and b uniquely in *lstsqr.m*. This can be generalized as follows: For the prediction of parameters of each state of a system uniquely, at least 3 data vectors are required corresponding to that state.

In one state of the system with p variables,

$$E(n+1) = \begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_p \end{bmatrix} + \begin{bmatrix} m_{11} & 0 & ... & 0 \\ 0 & m_{22} & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & m_{pp} \end{bmatrix} E(n),$$

we have $2p$ unknowns which are $m_{11}, m_{22}, ..., m_{pp}, b_1, b_2, ..., b_p$. So we need $2p$ linearly independent equations for finding the unknown parameters. Each matrix operation has p equations in itself. For the given first 2 data vectors $E(1), E(2)$ we have p equations defined as:

$$\begin{aligned} E_{1,2} &= b_1 + m_{11}E_{1,1} \\ E_{2,2} &= b_2 + m_{22}E_{2,1} \\ &... \\ E_{p,2} &= b_p + m_{pp}E_{p,1}. \end{aligned} \tag{5.2.6}$$

So we need 1 more matrix operation to conclude the equation number to $2p$. By using the data vectors $E(2), E(3)$, we have p more equations:

$$\begin{aligned} E_{1,3} &= b_1 + m_{11}E_{1,2} \\ E_{2,3} &= b_2 + m_{22}E_{2,2} \\ &... \\ E_{p,3} &= b_p + m_{pp}E_{p,2}. \end{aligned} \tag{5.2.7}$$

The solution of equations (5.2.6)-(5.2.7) allows us to find the unknowns of the system uniquely.

Hence, it can actually be concluded that for any multistate system, at least 3 data vectors of the corresponding state are required for the implementation of the Matlab function *lstsqr.m*.

In practice, the system of normal equations has two main drawbacks, loss of information and a condition squaring effect [29]. However, during this study this method proved itself applicable. Nevertheless other numerical methods for least squares problems could have been applied for the solution [27, 29].

5.3 Detection of Switching Times

In this part, multistate system is considered. According to the given data corresponding to more than one state, a Matlab function *hybd.m* (Appendix D) takes the data and a vector of *general error terms (GET)* of the variables as input. In the function, the provided data are partitioned into subdata by using *lstsqr.m* as a subfunction. The flow diagram of the function is as follows:

Step 1: Select first 3 data vectors and estimate the state transition matrix M and the state transition vector b by using *lstsqr.m*. Then, calculate the next data vector by using the transition matrix M and the transition vector b and compare this value with the original data at that time.

Step 2: If the difference between these two vectors is smaller than GET, then continue with the next data vector and repeat the operations defined in Step 1 until the difference is greater than GET.

Step 3: When the difference is greater than GET, conclude that the state transition boundary is reached. If this occurs at the n^{th} time sample, identify the state

transition matrix and the state transition vector according to the data vectors defined for that state. Take the n^{th} time sample as the beginning of the new state and repeat Step 1, Step 2, Step 3 until all the data are partitioned. At the end of these iterations available data are partitioned into subdata which correspond to one state of the system. The state transition matrices and state transition vectors that represent each partitioned data is also available.

Step 4: This step includes grouping the same states. The comparison criteria are according to the first determinants of the transition matrices of the partitioned data sets. If the determinants are found to be the same, then the eigenvalues of the matrices are compared. If the eigenvalues are also the same, finally, the state transition vectors of the corresponding matrices are compared. If the state transition vectors are found to be the same, then it is concluded that compared data sets belong to the same state.

Step 5: State transition matrices and state transition vectors of each detected state are decided by taking the mean of the transition matrices and transition vectors of the data sets that represents the same state. Finally, all possible state transition matrices and state transition vectors are inferred, however, it is still not known which state of the system these matrices and vectors represent. This remains to be a black box until the detection of the thresholds of the variables.

5.4 Threshold Detection

Correct decision of the system states is important for the inference of the system dynamics which depends on the detection of threshold values of each variable correctly.

A Matlab function *tresholds.m* (Appendix C) detects the thresholds of the variables defined in equation (4.6.1) by taking the data vectors corresponds to the switching

times using *hybd.m* as a subfunction. The flow diagram of *tresholds.m* is as follows:

Step 1: Group the data vectors of switching times according to their states.

Step 2: According to the assumptions discussed in Section 4.6, if a gene is regulated by an another gene, then only the row of the transition matrix or transition vector corresponding to that gene changes in the next state. Using this assumption, detect the regulated variable and assign a zero value to that variable in the vectors which were obtained in Step 1.

Step 3: Each row represents the values of the variables at switching times. It is assumed that[1] only one variable cross its threshold at the switching time. Under those circumstances it can be concluded that the value of the variable passing its threshold is going to be approximately the same in each group. To detect this variable, the means and standard deviations of each row of the groups are calculated. The Matlab routines *mean* and *std* are used for the calculation of mean and standard deviations of each group.

Step 4: We would like to select the variables that have the least deviation from the mean. In *tresholds.m*, a reasonable deviation percentage from the mean is asked to the user. If the standard deviation of the variable in a group is smaller than its mean times the deviation percentage and if it is unique in this group, then it is concluded that this variable has passed its threshold. The mean of the row of this group is assigned as the threshold of that variable. In the case where the defined deviation percentage is very low for the selection of the thresholds, the predefined percentage is multiplied by 2 in each repetition of the procedure.

Step 5: If more than one variable is found in Step 4 then repeat Step 4 for other groups until a unique variable is found.

[1]In fact there are multiple number of genes performing highly correlated patterns. In most of the analysis works, those highly correlated genes are clustered and represented by a single variable. This clustering work can be seen as a part of preprocessing work in a real world application and in inference work decorrelated data can be assumed.

Step 6: Once a threshold value for a variable is found then, there is no need to use the data vectors which share the same row with that variable. So assign zero to these row for each group. Under the assumption that only one variable can pass its threshold at a switching time, assign zero to the all columns of the corresponding group.

Step 7: Repeat Step 4, Step 5 and Step 6 until all the group elements are zero.

Step 8: After all thresholds are decided, assign the value 0 or 1 to the data vector according to the detected threshold levels. In the created matrix each column represents the state of the system corresponding to each data vector.

5.5 Numerical Example

In this section, a numerical example is given for a well understanding of the procedure followed in Section 5.3 and Section 5.4.

A multistationary system of 3 different genes and their expression levels at 740 time points are given in Table 5.1.

First of all, the given data are partitioned into subdata sets according to the steps defined in Section 5.3 by using least squares method. The *general error term (GET)* is defined to be as $avdata \times (5.10^{-5})$, where avdata is the average of all data in the system.

At the end of Step 3 we will have

$$MV = \begin{bmatrix} 0.99 & 0 & 0 & 0.99 & 0 & 0 & ... & ... & ... & 0.99 & 0 & 0 \\ 0 & 0.99 & 0 & 0 & 0.99 & 0 & .. & ... & ... & 0 & 0.99 & 0 \\ 0 & 0 & 0.99 & 0 & 0 & 0.99 & ... & ... & ... & 0 & 0 & 0.99 \end{bmatrix}_{3\times 42},$$

Table 5.1: Expression levels of the genes gene1, gene2, gene3 for 740 time samples.

	gene1	**gene2**	**gene3**
sample1-E(1)	2.0000	2.0000	2.0000
sample2-E(2)	1.9950	2.0200	2.0700
...	...	...	...
sample86-E(86)	1.7128	3.1488	6.0209
sample87-E(87)	1.7107	3.2373	6.0507
sample88-E(88)	1.7086	3.3250	6.0802
...	...	...	..
sample166-E(166)	1.5943	8.0785	7.6201
...	...	...	...
sample695-E(695)	2.3442	5.7050	6.0493
sample696-E(696)	2.3358	5.7680	6.0788
...	...	...	...
sample740-E(740)	2.0371	7.9952	7.1228

$$bV = \begin{bmatrix} 0.0150 & 0.0150 & ... & 0.0150 \\ 0.0400 & 0.1200 & ... & 0.1200 \\ 0.0900 & 0.0900 & ... & 0.0900 \end{bmatrix}_{3\times 14},$$

$$states = \begin{bmatrix} 1 & 86 & ... & 694 \\ 85 & 165 & ... & 740 \end{bmatrix}_{2\times 14},$$

$$sVend = \begin{bmatrix} 1.7128 & 1.5952 & ... & 2.0371 \\ 3.1488 & 8.0389 & ... & 7.9952 \\ 6.0209 & 7.6668 & ... & 7.1228 \end{bmatrix}_{3\times 14},$$

where MV is a general matrix that consists of the state transition matrices. We have 3 genes, so each (3×3)- submatrix of MV identifies the state transition matrices where some of these matrices represent the same states of the system. In the same way, each column of bV identifies the state transition vectors, where some of these vectors represent the same states of the system. As a result, each (3×3)-submatrix

and the columns of bV corresponding to that submatrix represent one state of the system. Each column of *states* matrix stands for the state intervals and each column of $sVend$ represents the data vector where the state transition boundary is reached.

We need to group state transition matrices and state transition vectors which denote the same states, according to the comparison criteria defined in Step 4. So we will have

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}_{6\times 14},$$

where each row of T represents one state of the system and each column stands for the column of the *states* matrix. If the state intervals were found to be the same, then they are denoted by 1 otherwise they are denoted by 0. So it is obvious that intervals {1-85, 364-421, 638-693}, {86-165, 422-469, 694-739}, {166-209, 470-502}, {210-276, 503-558}, {277-328, 559-605}, {329-363, 606-637} are represented by the same transition matrix and transition vector. We need state transition matrices and state transition vectors identifying each set. After Step 5 we have

$$mv = \begin{bmatrix} 0.99 & 0 & 0 & 0.99 & 0 & 0 & ... & ... & ... & 0.99 & 0 & 0 \\ 0 & 0.99 & 0 & 0 & 0.99 & 0 & .. & ... & ... & 0 & 0.99 & 0 \\ 0 & 0 & 0.99 & 0 & 0 & 0.99 & ... & ... & ... & 0 & 0 & 0.99 \end{bmatrix}_{3\times 18},$$

$$bv = \begin{bmatrix} 0.0150 & 0.0150 & 0.0150 & 0.0450 & 0.0450 & 0.0150 \\ 0.0400 & 0.1200 & 0.1200 & 0.1200 & 0.0400 & 0.0400 \\ 0.0900 & 0.0900 & 0.0300 & 0.0300 & 0.0300 & 0.0300 \end{bmatrix}_{3\times 6}.$$

Here, each (3×3)-submatrix of mv and corresponding column of bv characterizes one state. But it is still not known which states they represent. We need to follow

the steps defined in Section 5.4 for clarifying these states; this implies detection of threshold values of the variables.

Since we know which intervals represent the same states (i.e., {1-85, 364-421, 638-693}, {86-165, 422-469, 694-739}, {166-209, 470-502}, {210-276, 503-558}, {277-328, 559-605}, {329-363, 606-637}) we can group the data vectors at the state transition boundaries in these intervals.

Table 5.2: Grouped data vectors that correspond to switching times. Each group represents one state of the system.

E(85)	1.7149	3.1402	5.9908
E(421)	2.3375	5.5764	5.9763
E(693)	2.3613	5.6580	5.9894
...	...	...	...
E(637)	3.0122	6.9106	3.7751

According to Step 2, from the comparison of state transition vectors following each other, we have:

$$\begin{bmatrix} \overbrace{}^{1^{st}\text{ group}} & & & \overbrace{}^{2^{nd}\text{ group}} & & \overbrace{}^{3^{rd}\text{ group}} & & \overbrace{}^{4^{th}\text{ group}} & & \overbrace{}^{5^{th}\text{ group}} & & \overbrace{}^{6^{th}\text{ group}} & \\ 0 & 0 & 0 & 0 & 0 & 0.03 & 0.03 & 0 & 0 & 0.03 & 0.03 & 0 & 0 \\ 0.08 & 0.08 & 0.08 & 0 & 0 & 0 & 0 & 0.08 & 0.08 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.06 & 0.06 & 0 & 0 & 0 & 0 & 0 & 0 & 0.06 & 0.06 \end{bmatrix}.$$

Each row represents gene1, gene2, gene3, respectively. From that matrix we can conclude that for the first and fourth groups gene2 is regulated (passed its threshold), for the second and sixth groups gene3 is regulated, for the third and fifth groups gene1 is regulated. Possible regulators for the first and fourth group are gene1 and gene3, for the second and sixth group gene1 and gene2 and, finally, for the third and fifth groups possible regulators are gene2 and gene3.

Now, we consider which gene is the regulator of the predefined groups. For this aim

we will follow the Steps 3-7. The deviation percentage is taken as 0.002. So we got the following thresholds of the variables:

$$tresholds = \begin{bmatrix} 2.9857 \\ 7.9919 \\ 5.9855 \end{bmatrix}.$$

At Step 8, with the help of the thresholds, the system states and their orders are found as:

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

$$sequence = [1\ 2\ 3\ 4\ 5\ 6\ 1\ 2\ 3\ 4\ 5\ 6\ 1\ 2\ 3].$$

Each column of *systemstates* characterizes the possible system states, and in the *sequence* array each element represents the corresponding column number of *systemstates* matrix. So it can be concluded that the data which is dealt with defines a system with a periodic attractor of 1→ 2→ 3→ 4 →5→ 6 →1. System states, their transition matrices and transition vectors are given in Table 5.3.

Table 5.3: System states, their transition matrices and transition vectors.

states	**M**	**b**
000	$\begin{bmatrix} 0.99 & 0 & 0 \\ 0 & 0.99 & 0 \\ 0 & 0 & 0.99 \end{bmatrix}$	$\begin{bmatrix} 0.015 \\ 0.04 \\ 0.09 \end{bmatrix}$
001	$\begin{bmatrix} 0.99 & 0 & 0 \\ 0 & 0.99 & 0 \\ 0 & 0 & 0.99 \end{bmatrix}$	$\begin{bmatrix} 0.015 \\ 0.12 \\ 0.09 \end{bmatrix}$
011	$\begin{bmatrix} 0.99 & 0 & 0 \\ 0 & 0.99 & 0 \\ 0 & 0 & 0.99 \end{bmatrix}$	$\begin{bmatrix} 0.015 \\ 0.12 \\ 0.03 \end{bmatrix}$
010	$\begin{bmatrix} 0.99 & 0 & 0 \\ 0 & 0.99 & 0 \\ 0 & 0 & 0.99 \end{bmatrix}$	$\begin{bmatrix} 0.045 \\ 0.012 \\ 0.03 \end{bmatrix}$
110	$\begin{bmatrix} 0.99 & 0 & 0 \\ 0 & 0.99 & 0 \\ 0 & 0 & 0.99 \end{bmatrix}$	$\begin{bmatrix} 0.045 \\ 0.04 \\ 0.03 \end{bmatrix}$
100	$\begin{bmatrix} 0.99 & 0 & 0 \\ 0 & 0.99 & 0 \\ 0 & 0 & 0.99 \end{bmatrix}$	$\begin{bmatrix} 0.015 \\ 0.04 \\ 0.03 \end{bmatrix}$

CHAPTER 6

VALIDATION

6.1 Data Requirements of the Inference Algorithm

Multistationary nature of the genomic regulation has already been discussed . It is obvious that any information about a state which system has not exhibited can not be extracted. To be able to extract some predictive information, the observed system should experience a variety of different states and conditions. Not only for this particular algorithm, but almost for any inferential modeling procedure observations should fulfill some requirements. In this part those requirements of the observations for this model are discussed. Those requirements are analogous to the "*under which variety of conditions should experiments or observations shall be done*" problem of applied (nature) sciences.

As described in Section 3.2, the series of state transitions are called *trajectories*. The repetitive parts of the trajectories are called *attractors* and they can be one or more states long.

To find a unique solution for the threshold values of the genes, the trajectories of the system must be passed through at least twice with different entry values (until all the threshold values of the genes are defined).

There are at least two cases that the inference algorithm fails since the above condition does not hold. Firstly, if the system has an attractor of more than one state long (periodic attractor), then once the periodic attractor is reached, each system

state will be entered with the same concentration levels. Therefore, for a system of n genes $n-1$ regulators can be defined where their corresponding concentrations can be assigned as their threshold values. This prevents us from detecting the correct regulator for that switching time and, hence, its threshold value. Secondly, if the system has a single attractor, then, even though each system state can be visited once before the attractor state has been visited, the system will be trapped, preventing the above condition to hold. Thus, for a system of n genes only one concentration vector is detected at the switching times for each system states, and it is not sufficient to guess the threshold values of the genes from only one concentration vector.

On the other hand, when the above condition is satisfied so that each state has been visited more than once with different entry values, all the genes accept that the regulator will have as many different values as the number of visits to that state. This enables us to uniquely identify the regulator and its threshold value.

6.2 Validation of the Model

Once a mathematical model has been proposed, the evolution of the system can be simulated by solving the proposed model equations with suitable initial conditions. Then, the obtained results are compared to the behavior of the real system, and if the comparisons are satisfactory, the model can be regarded as a valid one, otherwise it is revised or rejected [6].

In the case where experimental data are not available, valid simulated data can be used instead. However, validation of the model strongly depends on the validity of the simulated data.

In order to determine the validity of the model, it has to be satisfied that the analyzed data and the calculated data are close to each other. Therefore, it can be

claimed that the mathematical model is valid if

$$\|cd - ad\| \leq \varepsilon \|ad\|,$$

where cd is the calculated data vector, ad is the analyzed data vector and ε is the maximum admissible gap related to the size of the analyzed data [6]. Additionally, $\frac{\|cd-ad\|}{\|ad\|}$ represents the relative error.

The selection of the norm should be based on the kind of approximation needed by the model. In this study, *averaged Euclidean* norm is used for an average error prediction

$$\|cd - ad\| = \left(\sum_{i=1}^{m} ((cd(i) - ad(i))^2) \right)^{\frac{1}{2}} /m, \tag{6.2.1}$$

where m is the number of samples and i represents the i^{th} entry of the data vector [6, 28].

6.3 Analysis of a Network with 6 Genes

For the validation of the model and the inference algorithm, an artificial data is used. The data are generated using an algorithm that is developed by Hakanoğlu and Öktem which is available in [63].

A system of 6 variables with a periodic attractor is generated where the system states are chosen as:

$$\begin{array}{ccccccccccc} 000000 & \rightarrow & 100000 & \rightarrow & 110000 & \rightarrow & 110100 & \rightarrow & 110110 & \rightarrow & 111110 \\ \uparrow & & & & & & & & & & \downarrow \\ 000001 & \leftarrow & 000011 & \leftarrow & 001011 & \leftarrow & 001111 & \leftarrow & 101111 & \leftarrow & 111111. \end{array}$$

The system has 1000 time samples (see Figure 6.1), where the threshold values and

the initial values are taken as

$$threshold = [3\ 5\ 7\ 9\ 4\ 6],$$

$$y_0 = [2\ 2\ 18\ 2\ 18\ 18],$$

for each variable.

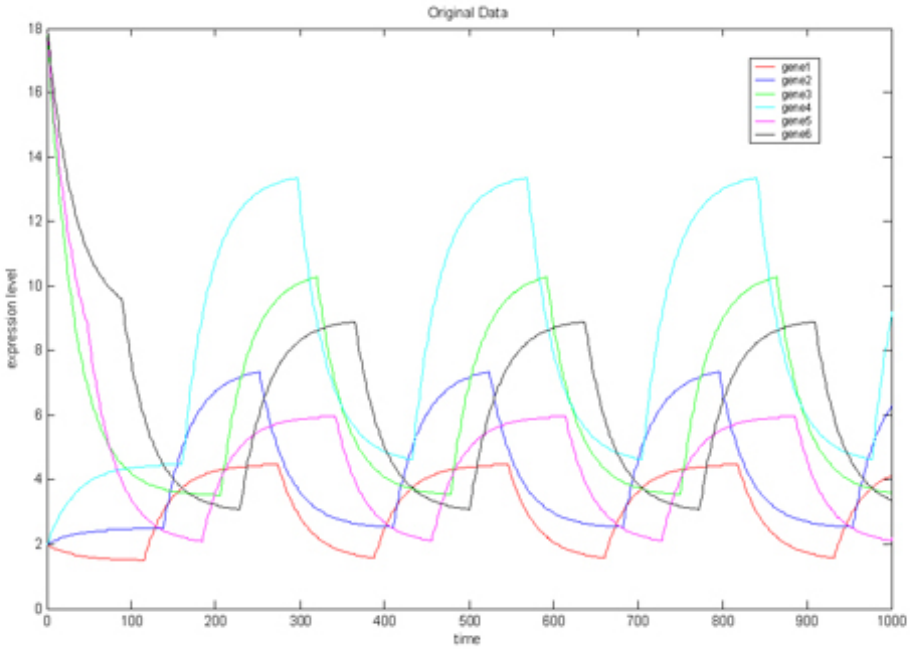

Figure 6.1: Expression data for 6 genes.

The phase space projections of the system are represented in Figure 6.2 where the red lines indicate the trajectories and the corresponding axes represent the expression levels. Additionally, the threshold values of each variable are plotted with dashed lines.

From the generated 1000 data, 590 are used for inference of the system parameters and the rest is used for comparison with the extrapolation results. The advantage of choosing a system with a periodic attractor is that the system cycles through each of its states enable a complete inference of the system states.

The inference algorithm is applied to the artificial data of 590 samples where GET (General error term) and deviation percentage are chosen as 10^{-4} and 0.002, respectively. The system states, corresponding state parameters and threshold values are found as:

$$M_{001011} = M_{000011} = M_{000001} = M_{000000} = M_{100000} = M_{110000} =$$

$$M_{110100} = M_{110110} = M_{111110} = M_{111111} = M_{101111} = M_{001111}$$

$$= \begin{bmatrix} 0.97 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.97 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.97 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.97 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.97 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.97 \end{bmatrix},$$

$$b_{001011} = \begin{bmatrix} 0.0450 \\ 0.0750 \\ 0.1050 \\ 0.1350 \\ 0.1800 \\ 0.2700 \end{bmatrix}, b_{000011} = \begin{bmatrix} 0.0450 \\ 0.0750 \\ 0.1050 \\ 0.1350 \\ 0.0600 \\ 0.2700 \end{bmatrix}, b_{000001} = \begin{bmatrix} 0.0450 \\ 0.0750 \\ 0.1050 \\ 0.1350 \\ 0.0600 \\ 0.0900 \end{bmatrix},$$

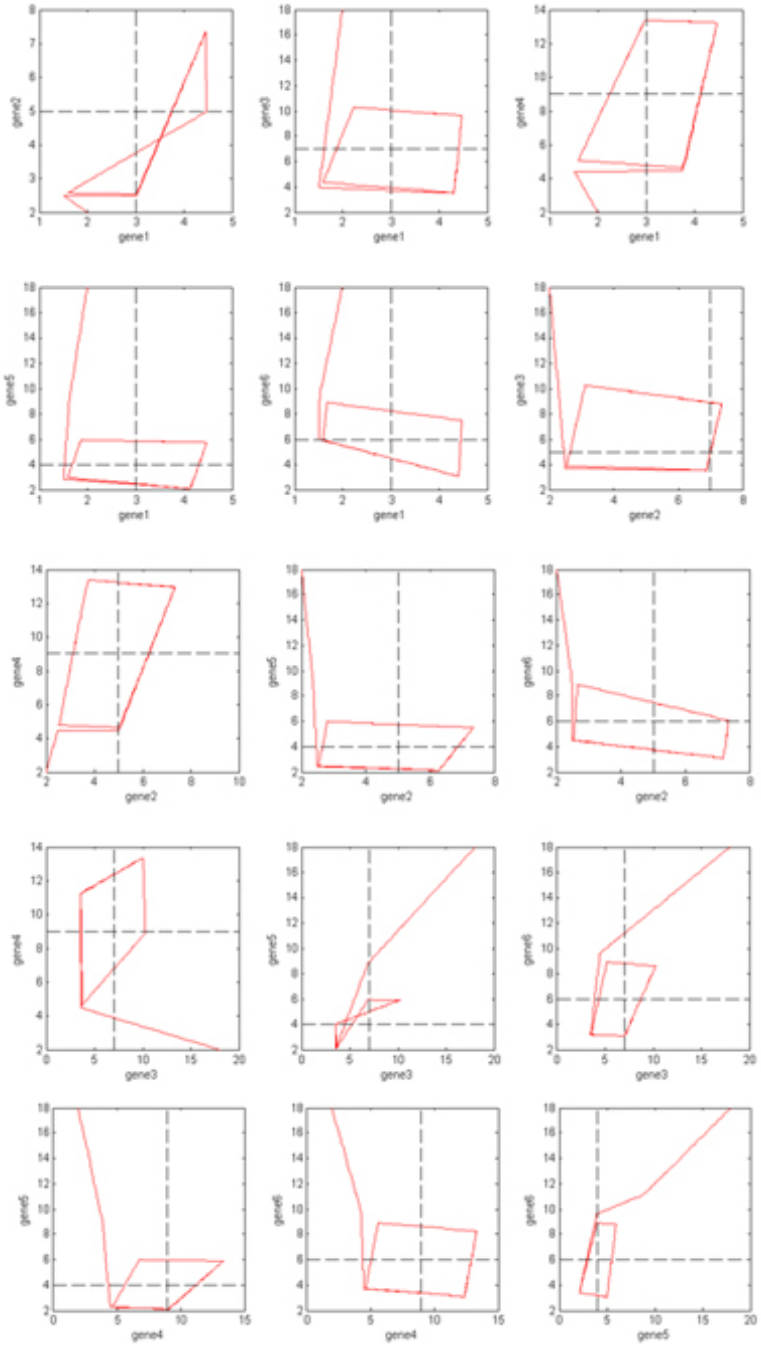

Figure 6.2: Phase space projections of the 6 genes network of Fig. 6.1.

$$b_{000000} = \begin{bmatrix} 0.1350 \\ 0.0750 \\ 0.1050 \\ 0.1350 \\ 0.0600 \\ 0.0900 \end{bmatrix}, b_{100000} = \begin{bmatrix} 0.1350 \\ 0.2250 \\ 0.1050 \\ 0.1350 \\ 0.0600 \\ 0.0900 \end{bmatrix}, b_{110000} = \begin{bmatrix} 0.1350 \\ 0.2250 \\ 0.1050 \\ 0.4050 \\ 0.0600 \\ 0.0900 \end{bmatrix},$$

$$b_{110100} = \begin{bmatrix} 0.1350 \\ 0.2250 \\ 0.1050 \\ 0.4050 \\ 0.1800 \\ 0.0900 \end{bmatrix}, b_{110110} = \begin{bmatrix} 0.1350 \\ 0.2250 \\ 0.3150 \\ 0.4050 \\ 0.1800 \\ 0.0900 \end{bmatrix}, b_{111110} = \begin{bmatrix} 0.1350 \\ 0.2250 \\ 0.3150 \\ 0.4050 \\ 0.1800 \\ 0.2700 \end{bmatrix},$$

$$b_{111111} = \begin{bmatrix} 0.1350 \\ 0.0750 \\ 0.3150 \\ 0.4050 \\ 0.1800 \\ 0.2700 \end{bmatrix}, b_{101111} = \begin{bmatrix} 0.0450 \\ 0.0750 \\ 0.3150 \\ 0.4050 \\ 0.1800 \\ 0.2700 \end{bmatrix}, b_{001111} = \begin{bmatrix} 0.0450 \\ 0.0750 \\ 0.3150 \\ 0.1350 \\ 0.1800 \\ 0.2700 \end{bmatrix},$$

$$threshold = [2.9696 \quad 4.9498 \quad 7.0749 \quad 8.9262 \quad 4.0306 \quad 6.0559]^T.$$

When the data are regenerated with the calculated system parameters and with the same initial values Figure 6.3 is obtained. In this figure, the solid lines represent the original data and the dashed ones represent the predictions.

The curves at the bottom of the graph represent the errors $|cd_j(i) - ad_j(i)|$, where $i = 1, 2, ..., 1000$ represents the time samples, $j = 1, 2, ..., 6$ represents the variables, cd is the calculated data vector and ad is the analyzed data vector. The average

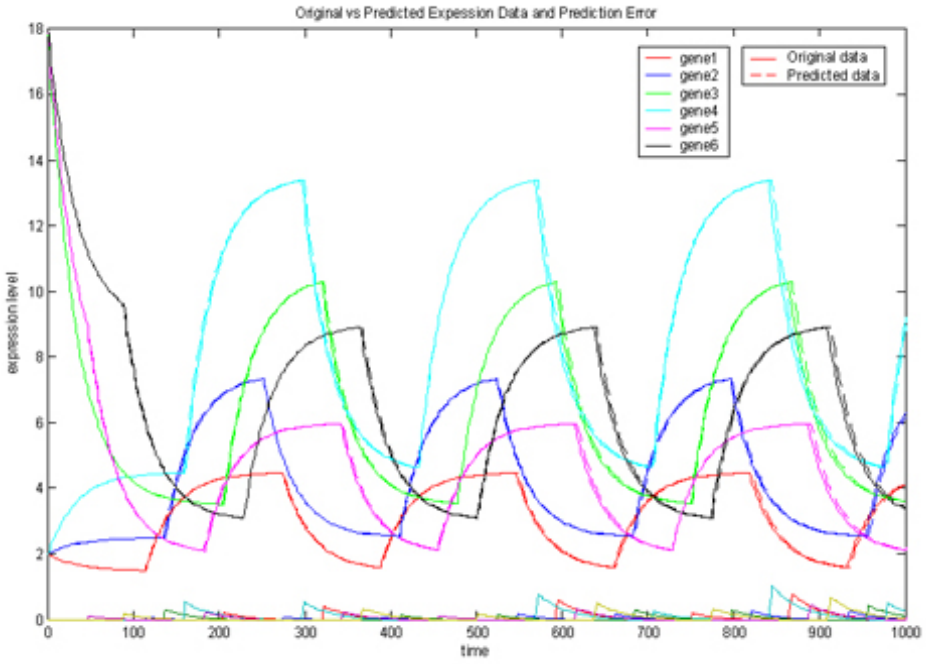

Figure 6.3: Original vs. predicted expression data and prediction errors.

error terms of each variable are calculated according to equation (6.2.1) as

$$averageerror = \begin{bmatrix} 0.0019 \\ 0.0029 \\ 0.0047 \\ 0.0066 \\ 0.0029 \\ 0.0043 \end{bmatrix}.$$

The calculated system parameters are sufficient to construct the network structure. From the parameters it is possible to get variable interactions as follows. During a state change, one of the variables undergoes a variation in behaviour resulting in different coefficients in its state equation. Simultaneously, another variable crosses its threshold which reflects itself in the Boolean array designating the system states.

Depending on the behaviour of the regulated variable it is inferred that the regulator either activates it or represses it, where *activation* means an increase in the rate of concentration and *repression* means a decrease in the rate. Based on this criterion the network structure is inferred as in Figure 6.4.

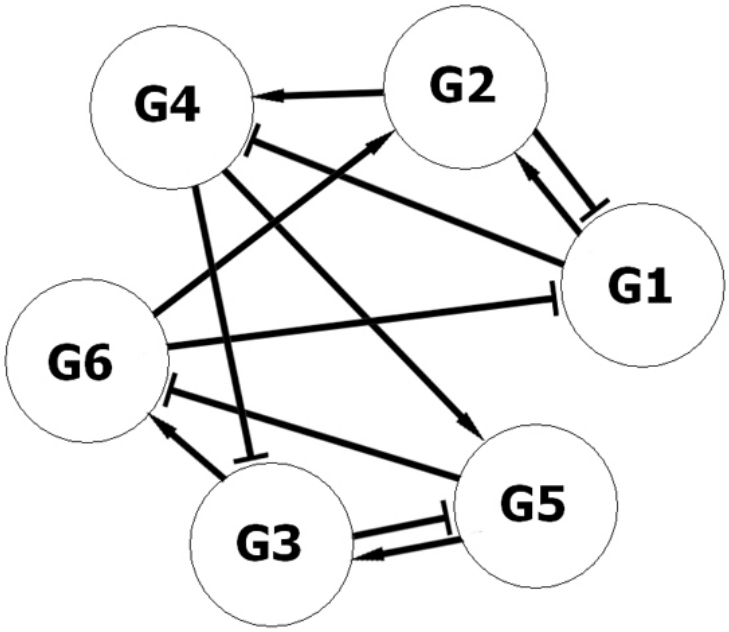

Figure 6.4: The network diagram of 6 genes. Activation indicated by $\rightarrow$, repression by $\dashv$.

In this figure, each variable has two regulators where one cause an increase and the other cause a decrease in its expression level.

In case of noisy data, it is difficult to determine the switching times. Even if they are detected, it is not possible to determine the system parameters, hence, the network structure. The reason is that very small fluctuations in the data cause significant differences in the state transition matrix and the state transition vector; thus it is not possible to find a unique representation for each state.

CHAPTER 7

DISCUSSION, CONCLUSION AND OUTLOOK

Most aspects of everyday's life are nonlinear. Whenever parts of a dynamical system interfere, or cooperate, or compete, there are nonlinear interactions going on, and most nonlinear systems are impossible to solve analytically [55]. Modeling of the regulatory mechanisms underlying dynamical systems can be approached under two different perspectives. When the detailed analysis of the system of interest is available, the first principle models generally lead to exact models of that system. Nevertheless, most of the time sufficient information to build first principle models is not available. In that case, the estimation of the underlying dynamics has to be carried out using the available empirical observations on the system which leads us to the inferential modeling approach.

The most critical challenge to realize is that each problem has its own characteristics and must be considered accordingly. Therefore, the selection of the application area is as fundamentally important as selecting the appropriate model for investigating the system dynamics.

In this study, a gene regulatory dynamics is selected as the application area. Genes control cellular processes by initiating protein synthesis through mRNA transcription. In order that a gene is transcribed into mRNA, it is necessary for a specific protein (transcription factors) to attach to the appropriate region of the DNA. Therefore, transcription factors are synthesized from mRNA transcription and these

transcription factors are used to start the mRNA transcription. Hence, this cycle defines a dynamical system involving highly nonlinear feedback mechanisms.

During the study, three main popular modeling approaches of gene networks are investigated, namely Boolean networks, differential equation models and piecewise linear equation models. It is possible to represent the complex behavior of the regulatory mechanisms underlying gene networks with Boolean networks. However, this type of networks is Boolean, as the name suggests, and synchronous. Due to these properties, the Boolean network approach has serious limitations in representing real systems. Linear differential equation models performed around a steady state yield good predictions. However, due to the nonlinear behavior of gene regulatory interactions, these models are also insufficient in supporting the dynamical variability of systems.

On the other hand, piecewise linear equation models, combine the capabilities of Boolean network and linear differential equation model approaches. Since genomic regulation can be treated as a multi-stationary system which may exhibit more than one possible stable states and can perform a transition from one stable state to another, the states are approximated by the Boolean network approach and the behavior in each stable state is approximated by linear differential equations. Consequently, piecewise linear equation models seem to provide a more realistic approach and it is well adopted to explain the gene regulatory mechanism.

After the above considerations, a piecewise linear formulation is selected to be used for the modeling of gene regulatory networks in this study. The formulation is described by a state transition matrix and a state transition vector with a Boolean function indicating the switching conditions. For the estimation of the model parameters and system states an inference algorithm is developed and the data requirements of the inference algorithm are emphasized. For the unique inference of the parameters M and b in each state, at least 3 data vectors are needed. Additionally, to find a unique solution for the threshold values of the genes, the trajectories of

the system must be passed through at least twice with different entry values. This shows that knowledge on the data plays a critical role for the inference of the gene networks. As a validation the algorithm is tested over an artificial 6 gene network and the network connections of each variable are clarified.

The developed algorithm, however, is based on a number of simplifying assumptions. The data with noise and time delays and, additionally, some special cases of the trajectory of the data make the network analysis difficult or impossible with the methods outlined in the thesis.

The main assumption of equation (4.6.1) is that the state transition matrices are diagonal. This means, the production rate of any gene within a state depends only on its own concentration. However, in reality there might exist other genes which have an effect on the production rate of the considered gene. This issue needs to be addressed before applying the inference algorithm to a real gene expression data.

Furthermore, from a biological point of view, the time delay in gene regulation arises from the actual delays characterizing the various underlying processes such as transcription, translation and transport [67]. Insertion of the time delays to the model will therefore make the model more realistic. Gene expression monitoring technology can not simultaneously measure the concentrations of all chemical species related to gene expression (mRNAs, proteins, small molecules, etc.). Hence, many methods being proposed for the network inference omit the influence of unobserved factors [48].

Different types of networks can also be explained using switching equations like for instance, the traffic networks. Traffic can be modeled around a junction with traffic lights. If the traffic light for the i^{th} road is green, then a normal freeway traffic exists, whereas, if it is red then an accumulation mode comes into consideration. The state of one road has a direct effect on the state of neighboring roads and an indirect effect on the states of the other roads. Thus, it is a system which comprises discrete (traffic lights) and continuous variables (flow and density) regulating each

other where the states of the traffic lights can be explained by Boolean functions and the flow on the roads can be explained by differential equations.

REFERENCES

[1] Aihara, K. and Chen, L., Stability of genetic regulatory networks with time delay, IEEE 49, 5 (2002) 602-608.

[2] Akutsu, T. and Miyano, S., Identification of genetic networks from a small number of gene expression patterns under the boolean network model, Proc. Pacific Symposium on Biocomputing 4 (1999) 17-28.

[3] Akhmet, M.U., Gebert, J., Öktem, H., Pickl, S.W. and Weber, G.W., An improved algorithm for analytical modeling and anticipation of gene expression patterns, Journal of Computational Technologies 9, 6 (2004) 40-48.

[4] Arkin, A. and McAdams, H.H., Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected Escherichia coli cells, Genetics 149 (1998) 1633-1648.

[5] Aster, R.C., Borchers, B. and Thurber, C.H., *Parameter Estimation and Inverse Problems*, Academic Press (2004).

[6] Bellomo, N. and Preziosi, L., *Modelling Mathematical Methods and Scientific Computation*, CRC Press, 1995.

[7] Brin, M. and Stuck, G., *Introduction to Dynamical Systems*, Cambridge University Press, 2002.

[8] Burden, R.L. and Faires, J.D., *Numerical Analysis*, Brooks/Cole, Thomson Learning Academic Resource Center, Seventh Edition, 2001.

[9] Chen, T., He, H.L. and Church, G.M., Modelling gene expression with differential equations, Pacific Symposium on Biocomputing, (1999) 29-40.

[10] Cinquin, O. and Demongeot, J., Roles of positive and negative feedback in biological systems, C.R. Biol. 325, 11 (2002) 1085-1095.

[11] De Jong, H., Modelling and simulation of genetic regulatory systems: A literature review, Journal of Comp. Biol. 9, 1 (2002) 67-103.

[12] Delbruck, M., In unites biologiques douees de continuite genetique, Editions du Centre National de la Recherche Scientifique, Paris, (1949) 33-35.

[13] D'haeseleer, P., *Reconstructing Gene Networks from Large Scale Gene Expression Data*, Doctor of Philosophy, Computer Science, The University of New Mexico, December 2000.

[14] D'haeseleer, P., Liang, S. and Somogyi, R., Gene expression data analyis and modeling, Tutorial, Pacific Symposium on Biocomputing, (1999).

[15] D'haeseleer, P., Liang, S. and Somogyi, Genetic network inference: From co-expression clustering to reverse engineering, Bioinformatics 16 (2002) 707-726.

[16] D'haeseleer, P., Wen, X., Fuhrman, S. and Somogyi, R., Linear modeling of mrna expression levels during cns development and injury, Pacific Symposium on Biocomputing, (1999).

[17] Edwards, R., Analysis of continuous time switching networks, Physica D 146 (2000) 165-199.

[18] Edwards, R. and Glass, L., Combinatorial explosion in model gene neworks, Chaos 10, 3 (2000) 691-704.

[19] Edwards, R., Siegelmann, H.T., Aziza, K. and Glass, L., Symbolic dynamics and computation in model gene networks, Chaos 11, 1 (2000) 160-169.

[20] Farcot, E. and Tournier, L., Hybrid model of gene regulatory networks, the case of lac-operon, Technical Report (2002).

[21] Filkov, V., *Gene Network Inference from Large-scale Expression Data*, CRC Press, LLC, 2001.

[22] Gebert, J., Latsch, S.W., Weber, G.W. and Wunschiers, R., An algorithm to analyze stability of gene expression patterns, to appear in the special issue *Discrete Mathematics and Data Mining* of Discrete Applied Mathematics 144-145, guest editors: E. Boros, P.L. Hammer and A. Kogan (2005).

[23] Gebert, J., Radde, N. and Weber, G.W., Modelling gene regulatory networks with piecewise linear differential equations, to appear in the special issue (feature cluster) *Challenges of Continuous Optimization in Theory and Applications* of European Journal of Operational Research (2006).

[24] Gebert, J., Latsch, S.W., Weber, G.W. and Wunschiers, R., Genetic networks and anticipation of gene expression patterns, invited paper, Computing Anticipatory Systems, CASYS'03 - Sixth International Conference, AIP Conference Proceedings 718 (2004) 474-485.

[25] Gouze, J.L., Positive and Negative Circuits in Dynamical Systems, Journal of Biological Systems 6, 1 (1998) 11-15.

[26] Gouze, J.L. and Sari, T., A class of piecewise linear differential equations arising in biological models, Dynamical Systems 17 (2002) 299-316.

[27] Hansen, P.C., Regularization Tools: A Matlab package for analysis and solution of discrete ill-posed problems, Numerical Algorithms 6, (1994) 1-35.

[28] Hastie, T., Tibshirani, R. and Friedman, J., *The Elements of Statistical Learning - Data Mining, Inference and Prediction*, Springer, 2001.

[29] Heath, M.T., *Scientific Computing, An Introductory Survey, The McGraw-Hill Companies Inc.*, Second Edition, 2002.

[30] Higham, D.J. and Higham, N.J., *Matlab Guide*, SIAM, 2000.

[31] Hoon, M., Imoto, S., Kobayashi, K., Ogasawara, N. and Miyano, S., Inferring gene regulatory networks from time ordered gene expression data of bacillus subtilis using differential equations, Pacific Symposium on Biocomputing (2003) 17-28.

[32] Hoon, M., Imoto, S. and Miyano, S., Inferring gene regulatory networks from time ordered gene expression data using differential equations, Lecture Notes in Computer Science 2534 (2002) 267-274.

[33] Hunt, S.P. and Livesey, F.J., *Functional Genomics*, Oxford University Press, 2000.

[34] Jong, H., D., Geiselmann, J., Batt, G., Hernandez, C. and Page, M., Qualitative simulation of the initiation of sporulation in bacillus subtilis, Bulletin of Mathematical Biology 66 (2004) 261-299.

[35] Jong, H., D., Gouze, J., L., Hernandez, C., Page, M., Sari, T. and Geiselmann, J., Qualitative simulation of genetic regulatory networks using piecewise-linear models, Bulletin of Mathematical Biology 66, 2 (2004) 301-40.

[36] Jong, H.D. and Page, M., Qualitative Simulation of Large and Complex Genetic Regulatory Systems, Proceeding of the 14th European Confernce on Artifical Intelligence ECAI 2000, IOS Press (2000) 141-145.

[37] Kobayashi, T., Chen, L., Aihara, K., Modeling genetic switches with positive feedback loops, J. theor. Biol. 221, (2003) 379-399.

[38] Lawson, C.L. and Hanson, R.J., *Solving Least Squares Problem*, Classics in Applied Mathematics 15, SIAM, 1995.

[39] Lee, M.T., *Analysis of Microarray Gene Expression Data*, Kluwer Academic Publishers, 2004.

[40] Madsen, K., Nielsen, H.B. and Tingleff, O., *Methods for Nonlinear Least Squares Problems*, Informatics and Mathematical Modelling, Technical Universty of Denmark, IMM, Second Edition, 2004.

[41] Mestl, T., *Mathematical Approaches to Describe and Analyze the Dynamics of Gene Regulatory Systems*, Dr. scient. thesis in mathematical biology, Agricultural University of Norway, July 1995.

[42] Monod, J. and Jacob, F., Genetic regulatory mechanisms in the synthesis of proteins, Journal of Molecular Biology 3 (1961) 318-356.

[43] Orians, G.H., *The Study of Life: An Introduction to Biology*, Boston, Allyn&Bacon 1969, Second Edition.

[44] Öktem, H., A survey on piecewise-linear models of regulatory dynamical systems, Nonlinear Analysis 63, 3 (2005) 336-349.

[45] Öktem, H., *Hybrid Systems*, Lecture Notes,

http://hybrid.iam.metu.edu.tr/iam570, (December 2005).

[46] Öktem, H., Research proposal for investigating genetic networks, Research Proposal, METU, Institute of Applied Mathematics.

[47] Öktem, H., Pearson, R. and Egiazarian, K., An adjustable aperiodic model class of genomic interactions using continuous time Boolean networks (Boolean delay equations), Chaos 13 (2003) 1167-1174.

[48] Perkins, T., H., Hallett, M. and Glass, L., Inferring models of gene expression dynamics, Journal of Theoretical Biolology 230 (2004) 289-299.

[49] Prestwich, K.N., *Genes, Proteins, and Performance: The Nature of Poteins*, Lecture Notes, Dept. of Biology, College of the Holy Cross, (2003).

[50] Sakamoto, E. and Iba, H., Inferring a system of differential equations for a gene regulatory network by using genetic programming, Congress on Evolutionary Computation, CEC 2001.

[51] Sari, T. and Gouze, J., A class of piecewise-linear equations arising in biological models, Dynamical Systems 17 (2003) 299-316.

[52] Shamir, R., *Analysis of Gene Expression Data*, Lecture Notes, 2002.

[53] Soule, C., Graphic requirements for multistationarity, ComPlexUs 1 (2003) 123-133.

[54] Stark, J., Brewert, D., Barenco, M., Tomescut, D., Callard, R. and Hubank, M., Reconstructing gene networks: what are the limits?, Biochemical Society 31, 6 (2003) 1519-1525.

[55] Strogatz, S.H., *Nonlinear Dynamics and Chaos*, Addison-Wesley Publishing Company, 1994.

[56] Styczynski, M., P. and Stephanopoulos, G., Overview of computational methods for the inference of gene regulatory networks, Computers and Chemical Engineering 29 (2005) 519-534.

[57] Takane, M., *Inference of Gene Regulatory Networks from Large Scale Gene Expression Data,* School of Computer Science McGill University 2003.

[58] Taştan, M., *Analysis and Predictions of Gene Expression Patterns by Dynamical Systems, and by a Combinatorial Algorithm*, MSc thesis, Institute of Applied Mathematics, Middle East Technical University, August (2005).

[59] Tegner, J., Yeung, S., Hasty, J. and Collins, J., Reverse engineering gene networks: Integrating genetic perturbations with dynamical modelling, PNAS 100, 10 (2003) 5944-5949.

[60] Thomas, R., Laws for the dynamics of regulatory networks, Int. J. Dev. Biol. 42 (1998) 479-485.

[61] Thomas, R. and Kaufman, M., Multistatonarity, the basis of cell differentiation and memory. I. Structural conditions of multistationarity and other nontrivial behavior, Chaos 11, 1 (2001) 170-179.

[62] Wall, M.E., Hlavacek, W.S. and Savageau, M.A., Design of gene circuits: Lessons from bacteria, Nature 5 (2004) 34-42.

[63] Modeling multistationary processes by using hybrid system formulation: A study with priority on functional genomics homepage, http://hybrid.iam.metu.edu.tr/tubitak (December 2005).

[64] Numerical Recipes in C: The art of scientific computing homepage, http://www.nr.com (December 2005).

[65] Support Vector Machine Classification of Microarray Gene Expression Data homepage,

http://www.soe.ucsc.edu/research/compbio/genex/genexTR2html/ (December 2005).

[66] White, R.J., *Gene Transcription Mechanisms and Control*, Blackwell Science Ltd., 2001.

[67] Wu, F.X., Zhang, W-J. and Kusalik, A.J., State space modelfor gene regulatory networks with time delays, IEEE Computational Systems Bioinformatics Conference, (2004).

[68] Wuensche, A., Genomic regulation modeled as a network with basins of attraction, Pacific Symposium on Biocomputing (1998) 89-102.

[69] Yeung, S., Tegner, J. and Collins, J., Reverse engineering gene networks using singular value decomposition and robust regression, PNAS 99, 9 (2002) 6163-6168.

[70] Yılmaz, F.B., *A Mathematical Modelling and Approximation of Gene Expression Patterns by Linear and Quadratic Regulatory Relations and Analysis of Gene Networks*, MSc thesis, Institute of Applied Mathematics, Middle East Technical University, August 2004.

APPENDIX A

Linear Equations

The fundamental tool for solving a linear system of equations in Matlab is the backslash operator, \. It handles three types of linear system $Ax = b$, where the matrix A and the vector b are given. The three possible shapes for A lead to square, overdetermined and underdetermined systems [30].

A.1 Square Systems

If A is an *n-by-n* nonsingular matrix then $A\backslash b$ is the solution x to $Ax = b$, computed by *LU factorization with partial pivoting [30].*

For solving a linear system of the form $Ax = b$, matrix factorization can be used. The factorization is particularly useful when it has the form $A = LU$, where L is lower triangular and U is upper triangular. If A has been factored into the triangular form $A = LU$, then x can be solved by using a two step process. First $y = Ux$ is formed and then $Ly = b$ is solved for y. Not every matrix can be factorized in this way, but when row interchanges are incorporated, the factorization always exists [29]. Since L is triangular determining y from this equation requires $O(n^2)$ operations. Once y is known, the upper triangular system $Ux = y$ requires additional $O(n^2)$ operations to determine the solution x [8].

A.2 Overdetermined System

If A has dimension *m-by-n* with $m > n$ then $Ax = b$ is an *overdetermined system* that means, there are more equations than unknowns. In general, there is no x satisfying the system. Matlab's $A\backslash b$ gives a least squares solution to the system, that is it minimizes the Euclidean norm of the residual $r = b - Ax$ over all vectors x. If A has full rank n, there is a unique least squares solution. If A has rank k

less than n then $A\backslash b$ is a basic solution one with at most k nonzero elements (k is determined and x computed, using the *QR factorization with column pivoting*) [30]. A QR factorization of an *m-by-n* matrix A is a factorization $A = QR$, where Q is *m-by-m* unitary and R is *m-by-n* upper triangular [29].

A QR factorization with column pivoting has the form $AP = QR$ where P is a permutation matrix. The permutation strategy that is used produces a factor R whose diagonal elements are nonincreasing: $|r_{11}| \geq |r_{22}| \geq ... \geq |r_{nn}|$ [29]. Column pivoting is particularly appropriate when A is suspected of being rank deficient, as it helps to reveal near rank deficiency [30].

A.3 Underdetermined System

If A has *m-by-n* dimension with $m < n$ then, $Ax = b$ is an underdetermined system that means, there are fewer equations than unknowns. The system has either no solution or infinitely many.

APPENDIX B

leastsquares.m

```
% (C) Didem AKÇAY (December, 2005)
function [M,b,LMS] = leastsquares(data)
% Least squares method for multidimensional data
% Data vector, say vector "y"
% satisfies y(m+1) = M * y(m) + b property
% m is the time sample
%
% leastsquares(data) finds M, b that best defines the above property
% for the given data
%
%
[n,m]=size(data);
datanew=[ones(1,m);data];
for i=1:n+1
    for j=1:n+1
        K(i,j)=0;
            for k=1:m-1
                K(i,j)=K(i,j)+datanew(i,k).*datanew(j,k);
            end
    end
for l=1:n
    F(i,l)=0;
        for k=1:m-1
```

```
            F(i,l)=F(i,l)+datanew(l+1,k+1).*datanew(i,k);
        end
end
end
for i=1:n
    p(:,i)=K\F(:,i);
end
u=p';
M=u(:,2:n+1);
b=u(:,1);
```

APPENDIX C

tresholds.m

```
% (C) Didem AKÇAY (December, 2005)
function [mv,bv,states,sequence,treshtresh]=tresholds(data,dp,GET)
%
% detection of the threshold values of the variables
% dp represents the deviation percentage from the mean
% GET represents the general error term
%
[temp22,temp,T,mv,bv]=hybrid(data,GET);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% detection of the regulated variable
nT=size(T);nTmain=nT;
tempnew=temp22;
[nend,mend]=size(tempnew);
for i=1:nend
    for j=1:mend
        if tempnew(i,j)<=10^-4
            tempnew(i,j)=1;
        else
            tempnew(i,j)=0;
        end
    end
end
```

```
tempend=temp.*tempnew; % possible variables and their values that can pass thresh-
old
tdene=cumsum(sum(T(:,1:nTmain(2)-1)')); % cumulative sum of the repetitive states
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tcum=tdene;
tempwork=tempend;
i=0;
error=dp;
treshtresh=zeros(size(tempwork,1),1);
Tcum;
while (max(max(tempwork))>0) && (error < 100)
% iterations are done until all the elements of
% tempwork is going to be zero
% maximum error rate defined for that
% iterations are 100
error=error*2;
% at each iteration error rate is inreased
% work includes the goups
    if (i==0)
        work=tempwork(:,1:(Tcum(i+1)));
    else
        work=tempwork(:,(Tcum(i)+1):(Tcum(i+1)));
    end
%
% mean and standart deviation of each group is calculated
```

```
    for p=1:size(work,1)
        workstd(p)=(std(work(p,:)));
        workmean(p)=(mean(work(p,:)));
    end
workstd=workstd';
workmean=workmean';
mincount=0;
variables=[];
% For each variable of the same group if their standart deviation is
% smaller than their mean*errorrate
% and also this is true for only one variable in the group
    for variable=1:size(tempwork,1)
        if ( (workstd(variable)<workmean(variable)*error) && (workmean(variable)~=0)
)
            mincount=mincount+1; % counts the elements of variables vector
            variables=[variables;variable]; % collect them in a vector
        end
    end
%
% then the mean of the values corresponding to that
% variable can be defined as a threshold for that variable
    if (mincount==1)
        treshtresh(variables)=workmean(variables);
        % closer values for the threshold value in tempwork corresponding
        % to that variable
        % delete the closer values
        % sharing the same row with that variable
```

```
        for column=1:size(tempwork,2)
            if (abs(treshtresh(variables)-tempwork(variables,column))...
            <workmean(variables)*error)
                tempwork(:,column)=0;
            end
        end
        tempwork(variables,:)=0;
    end
% operations are done since all the threshold values are calculated
    i=mod(i+1,size(Tcum,2));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% System states and their order
[states,sequence]=syssts(data,treshtresh);
```

APPENDIX D

hybrid.m

```
% (C) Didem AKÇAY (December, 2005)
function [temp22,temp,T,mv,bv]=hybrid(data,GET)
%
% GET is a vector of general error term
% Dimension of GET must be same with the variable number
%
[n,m]=size(data);
%
% decision of the error limit
% evaluated as the average of the data
% times GET
avdata = sum(data')/m;
erlim = avdata.*(GET);
erlim = erlim';
s=1; p=0;
% MV holds state matrices
% bV holds state vectors
% sV holds vectors where state change occurs
% sv holds switching times where state change occurs
MV=[]; bV=[]; sV=[data(:,1)]; sv=[1];sve=[];sVend=[];
while p < m-2
p = p+1;
    if (p >= s+2) % lstsqrs.m method needs 3 data to run
```

```
            [M,b] = lstsqrs(data(:,s:p));
            ytrial=hybridsupport ( data(:,s), s, p+1, M, b);
            yreal=data(:,p+1);
            y=abs(yreal-ytrial)>=erlim;
                if sum(y)~=0
                    MV = [MV,M]; bV = [bV,b];
                    s=p;
                    sV = [sV,data(:,s)]; sVend=[sVend,data(:,s-1)];
                    sv = [sv,s];sve=[sve,s-1];
                end
        end
        if p==m-2
            [M,b] = lstsqrs(data(:,s:p+2));
            sve=[sve,m];MV = [MV,M]; bV = [bV,b]; sVend=[sVend,data(:,m)];
        end
end
states=[sv;sve];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% decide which states are same according to M and b
% and collect them in a matrix T
%
%
[n1,m1]=size(MV);
h=m1/n1;T=zeros(h,h);
for i=1:h
    T(i,i)=1;
        for j=i+1:h
```

```
                a=MV(:,((i-1)*n1)+1:i*n1);
                b=MV(:,(j-1)*n1+1:j*n1);
                if (abs(det(a)-det(b))<=10^-5)
                    if (abs(eig(a)-eig(b))<=10^-5)
                        if sum(abs(bV(:,i)-bV(:,j))<=10^-4*ones(n,1))==n
                            T(i,j)=1;T(j,i)=1;
                        end
                    end
                end
            end
end
TT=T;
%
% Elimination of the same rows of T
c=[];
for i=1:h-1
    for j=i+1:h
        if sum(T(i,:)==T(j,:))==h
            c=[c,j];
        end
    end
end
if isempty(c)==1
    T=TT;
else
    k=size(c);
    c=sort(c);
```

```
    d=c(1);
        for i=1:k(:,2)
            if d(length(d))~=c(i)
                d=[d c(i)];
            end
        end
        for i=length(d):-1:1
            T(d(i),:)=[];
        end
end
nT=size(T);nTmain=nT;
nT=nT(1);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% collect the difference of switching matrices in temp1
% collect the difference of switching vectors in temp2
% collect the transition vectors in temp3
%
temp11=[];temp22=[];temp=[];mv=[];bv=[];
for i=1:nT
    temp1=[]; temp2=[];temp3=[];mv2=[];bv2=[];
        for j=1:h-1
            if T(i,j)==1
                if temp2
                    % collection of state switching values of same states
                    temp3=[temp3,data(:,states(2,j))];
                      % collection of state difference matrices of same states
```

```
                temp1=[temp1,abs(MV(:,(j-1)*n+1:j*n)-MV(:,j*n+1:(j+1)*n))];
                % collection of state difference vectors of same states
                temp2=[temp2,abs(bV(:,j)-bV(:,j+1))];
                mv2=(mv2+MV(:,(j-1)*n+1:j*n))./2;
                bv2=(bv2+bV(:,j))./2;
            else
                temp3=data(:,states(2,j));
                temp1=abs(MV(:,(j-1)*n+1:j*n)-MV(:,j*n+1:(j+1)*n));
                temp2=abs(bV(:,j)-bV(:,j+1));
                mv2=MV(:,(j-1)*n+1:j*n);
                bv2=bV(:,j);
            end
        end
    end
    temp11=[temp11,temp1];% collection of state switching values of different states
    temp22=[temp22,temp2]; % collection of state difference matrices of different
states
    temp=[temp temp3]; % collection of state difference vectors of different states
    mv=[mv mv2];
    bv=[bv bv2];
end
```

APPENDIX E

lstsqrs.m

```
% (C) Didem AKÇAY (December, 2005)
function [M,b]=lstsqrs(data)
% Least squares method for multidimensional data
% Data vector, say vector "y"
% satisfies y(m+1) = M * y(m) + b property
% m is the time sample
%
% lstsqrs(data) finds M, b that best defines the above property
% for the given data
% M is supposed to be a diagonal matrix
%
[n,m]=size(data);
datanew=[ones(1,m);data];
datanew1=datanew;datanew2=datanew;
P=[];
m1=m;
m2=m;
while m1>=2
    for i=1:2
        for j=1:2
            K(i,j)=0;
                for k=1:m-1
                    K(i,j)=K(i,j)+datanew1(i,k).*datanew1(j,k);
```

```
                end
            end
        end
        P=[P K];
        datanew1(2,:)=[];
        [m1 n1]=size(datanew1);
end
while m2>=2
        for i=1:n
            for j=1:2
                F(i,j)=0;
                    for k=1:m-1
                        F(i,j)=F(i,j)+datanew2(2,k+1).*datanew2(j,k);
                    end
            end
            datanew2(2,:)=[];
            [m2 n2]=size(datanew2);
        end
end
F=F';
[ne me]=size(P); h=me/ne;c=[];
for j=1:h
        d=P(:,j*ne-1:j*ne+ne-2)\F(:,j);
        c=[c,d];
end
M=diag(c(2,:));
b=c(1,:);b=b';
```

APPENDIX F

syssts.m

```
% (C) Didem AKÇAY (December, 2005)
function [states,sequence]=syssts(data,threshold)
%
%syssts.m finds system states according to threshold values
%
[n,m]=size(data);
for i=1:n;
    for j=1:m
        if data(i,j)>=threshold(i)
            sts(i,j)=1;
        else
            sts(i,j)=0;
        end
    end
end
states=[];
for i=1:m
    fnd=0;
        for j=1:size(states,2)
            if sum(sts(:,i)==states(:,j))==n
                fnd=1;
            end
        end
```

```
        if fnd==0
            states=[states,sts(:,i)];
        end
end
sequence=[];
for i=1:m
    for j=1:size(states,2)
        if sum(sts(:,i)==states(:,j))==n
            if size(sequence)==0
                sequence=[sequence,j];
            end
            if sequence(size(sequence,2))~=j
                sequence=[sequence,j];
            end
        end
    end
end
```

APPENDIX G

hybridsupport.m

```
% (C) Didem AKÇAY (December, 2005)
function [y_final]=hybridsupport (y_initial, initial_sample_number, final_sample_number,
M, b)
%
% hybridsupport.m finds proceeding time samples
% according to the predefined function parameters
y_last=y_initial;
for current_step=initial_sample_number+1:final_sample_number
    y_new=M*y_last+b;
    y_last=y_new;
end
y_final=y_new;
```