University of Tennessee, Knoxville

TRACE: Tennessee Research and Creative Exchange

Doctoral Dissertations

Graduate School

12-2010

Differential Equation Models and Numerical Methods for Reverse Engineering Genetic Regulatory Networks

Mi Un Yoon
University of Tennessee, myoon@utk.edu

Follow this and additional works at: https://trace.tennessee.edu/utk_graddiss

 Part of the Numerical Analysis and Computation Commons

Recommended Citation

Yoon, Mi Un, "Differential Equation Models and Numerical Methods for Reverse Engineering Genetic Regulatory Networks. " PhD diss., University of Tennessee, 2010.
https://trace.tennessee.edu/utk_graddiss/928

This Dissertation is brought to you for free and open access by the Graduate School at TRACE: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of TRACE: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

To the Graduate Council:

I am submitting herewith a dissertation written by Mi Un Yoon entitled "Differential Equation Models and Numerical Methods for Reverse Engineering Genetic Regulatory Networks." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mathematics.

Xiaobing Feng, Major Professor

We have read this dissertation and recommend its acceptance:

Suzanne Lenhart, Steven Wise, Albrecht von Arnim

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by Miun Yoon entitled "Differential Equation Models and Numerical Methods for Reverse Engineering Genetic Regulatory Networks". I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mathematics.

Xiaobing Feng

Major Professor

We have read this dissertation
and recommend its acceptance:

Suzanne Lenhart

Steven Wise

Albrecht von Arnim

Accepted for the Council:

Carolyn R. Hodges

Vice Chancellor and Dean of
Graduate Studies

(Original signatures are on file with offical student records.)

Differential Equation Models and Numerical Methods for Reverse Engineering Genetic Regulatory Networks

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Miun Yoon

December 2010

Copyright © 2010 by Miun Yoon.

All rights reserved.

Dedication

This dissertation is dedicated to my family. My parents raised me with freedom of choice and supported me all the time. My parents-in-law encouraged and are proud of me.

I especially dedicate this dissertation to my husband, Kyungyuk, who has always been there as a husband, a mentor, and a friend, to my lovely son, Hagen, who has grown into a wonderful four years old boy and makes me smile and to my lovely daughter, Ella, who was born on October 13, 2010.

Acknowledgments

I would like to take this opportunity to thank many people who have made this dissertation possible. First and foremost, I would like to thank my dissertation advisor, Dr. Xiaobing Feng for his watchful guidance, kind encouragement, forgiveness. Dr. Feng has led me to this very fun field of study and never got tired of answering my questions. I would also like to thank Drs. Suzanne Lenhart, Steven Wise, and Albrecht von Arnim, for gladly serving on my dissertation committee. Dr. Suzanne Lenhart who gives me suggestions and great support as a woman mathematician. Dr. Wise gives me detailed comments and suggestions about numerical methods. Dr. von Arnim strengthens my background of Genetics and gives me a lot of suggestions as a biologist.

I would like to thank Drs. Jizhong Joe Zhou and Zhili He of Institute for Environmental Genomics (IEG) at University of Oklahoma, and Dr. Yunfeng David Yang of Oak Ridge National Laboratory for getting me interested in modeling and simulation of gene regulatory netwokrs, and system biology in general.

A number of people whom I have met at the UTK deserve my special thanks. Allison Carter, my longest buddy in this country, has been keeping me motivated and encouraged when I could not see the light at the end of the tunnel.

Finally, I would like to acknowledge the support of the NSF grants DMS-0410266 and DMS-0710831 for my research from August of 2008 to July of 2009 and in the summer of 2010.

Abstract

This dissertation develops and analyzes differential equation-based mathematical models and efficient numerical methods and algorithms for genetic regulatory network identification. The primary objectives of the dissertation are to design, analyze, and test a general variational framework and numerical methods for seeking its approximate solutions for reverse engineering genetic regulatory networks from microarray datasets using the approach based on differential equation modeling. In the proposed variational framework, no structure assumption on the genetic network is presumed, instead, the network is solely determined by the microarray profile of the network components and is identified through a well chosen variational principle which minimizes a biological energy functional. The variational principle serves not only as a selection criterion to pick up the right biological solution of the underlying differential equation model but also provides an effective mathematical characterization of the small-world property of genetic regulatory networks which has been observed in lab experiments. Five specific models within the variational framework and efficient numerical methods and algorithms for computing their solutions are proposed and analyzed in the dissertation. Model validations using both synthetic network datasets and real world subnetwork datasets of *Saccharomyces cerevisiae* (yeast) and *E. coli* are done on all five proposed variational models and a performance comparison vs some existing genetic regulatory network identification methods is also provided. As microarray data is typically noisy, in order to take into account the noise effect in the mathematical models, we propose a new approach based on stochastic differential equation modeling and generalize the deterministic variational framework to a stochastic variational framework which

relies on stochastic optimization. Numerical algorithms are also proposed for computing solutions of the stochastic variational models. To address the important issue of post-processing computed networks to reflect the small-world property of underlying genetic regulatory networks, a novel threshholding technique based on the Random Matrix Theory is proposed and tested on various synthetic network datasets.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background of genetics

This section collects some basic materials about genes, DNA, RNA, gene expression, and gene regulatory networks. These materials can be found in any undergraduate level text books on genes and gene networks, among them we refer to [46, 84].

1.1.1 History of genetics

Gregor Johann Mendel who is known as the "father of modern genetics" suggested the existence of inherited characteristics by pea plants in 1866. In 1902, Walter Sutton and Theodor Boveri established the chromosome theory of inheritance. In 1909, Wilhelm Johannsen who was a Danish Botanist named a functional unit of heredity as "gene". In the following years, Thomas Hunt Morgan who was an American geneticist and embryologist confirmed the chromosome theory of inheritance by his discovery that genes are carried by and located on specific part of a chromosome. Biochemists proved that each chromosome is an organized structure of a single piece of coiled DNA and DNA-bound proteins. In the 1940s and 1950s, Oswald Avery, Colin Munro MacLeod, and Maclyn McCarty finally demonstrated that genes are packed and encoded in DNA. This discovery served as the springboard for research on the DNA molecule to identify its structure and to understand

how DNA carries the hereditary information. In 1953, James D. Watson and Francis Crick proposed the structure of DNA which is known as the "double helix" these days [81]. After the discovery of the structure of DNA, it was also elucidated that the double helix structure of DNA and basepairs of four chemical components on the structure were related to the ability of DNA as the main storage and carrier of the genetic information.

Each gene is manual that encode the information to produce proteins and other molecules in the cell. To produce needed proteins, first, the information in each gene is read by a special molecule machinery in the cell. Second, those instructions are transferred to another molecule machine, the ribosome, to produce proteins using instructions. These two processes are referred as "gene expression". Francis Crick in 1958 proposed "the central dogma" in the cell (Figure 1.1). The first process is called transcription and the latter one is called translation. See the next subsection for a detailed discussion.

The biologists turned their attention to categorizing and analyzing the information contained within individual genes, which was the beginning of molecular genetics. More recently, the entire hereditary information contained within one species, the genome, has been analyzed (genomics). In 1990, the human genome project began to identify all the genes (about 20,000 - 25,000 genes) in human DNA by determining the entire sequence of basepairs in the human genome, which was completed in 2003.

Although all the genes in human DNA are now known, it is still hard to explain how living organisms are made up and how they respond to and adjust to their environment. In other words, to understand the metabolism and cellular functions, one needs to understand not only each component (gene) but also how it is expressed and how different genes interact. The development of DNA microarray technology has accelerated geneticists' research. The DNA microarray has allowed geneticists to examine the RNA expression of thousands of genes simultaneously. Using DNA microarray, geneticists have been able to determine when and where each gene is expressed, as well as how the expression of one gene depends on the presence or expression of other genes.

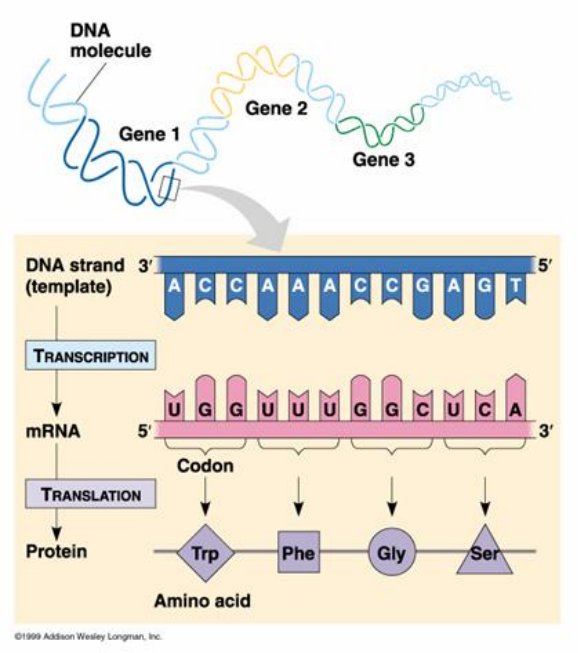


Figure 1.1: Central dogma of biology (http://1mkturin.files.wordpress.com)

1.1.2 Molecular basis of gene expression

Cells are the building blocks of the living organism. Each cell in a living organism consists of molecular components such as DNA, RNA, proteins, and other molecules. Cells mainly produce varieties of proteins which are needed for the development and functioning of living organisms. There are many different types of cells that differ in their molecular composition, structure, and function. For example, some cells in the blood produce hemoglobin to carry oxygen to every place in the body and other cells produce enzymes like amylase, pepsin, and lactase to digest food. Who determines which proteins cells produce and what the cells' functions are? The pattern of proteins produced b any given cell is determined by the genes encoded in the basepair sequence of the DAN and by the expression pattern of those genes (Figure 1.2).

DNA stores genetic information containing all instructions needed for cells to produce proteins and to function in the living organism. The double helix structure of DNA is

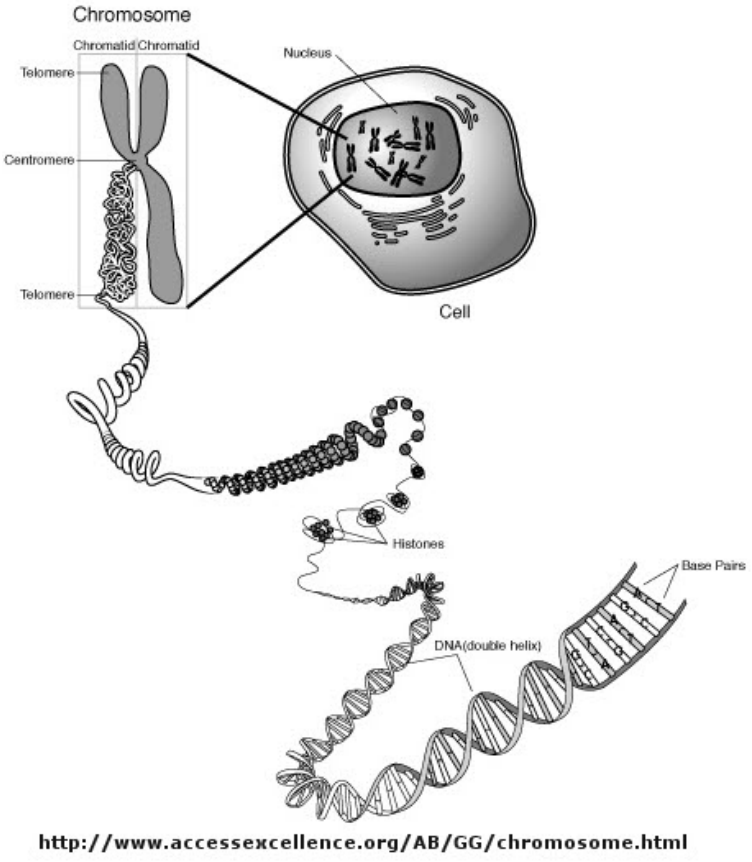


Figure 1.2: Cell, Chromosome, and DNA (http://employees.csbsju.edu)

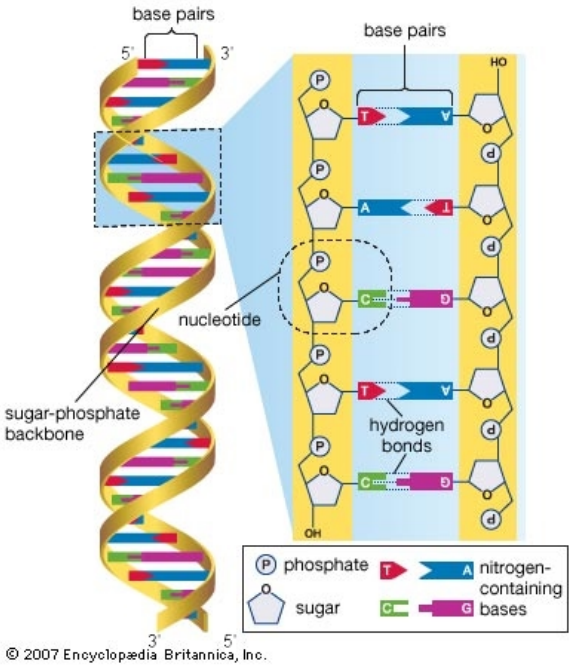


Figure 1.3: The structure of DNA, A: Adenin, C:Cytosine, G:Guanine, and T:Thymine (http://www.britannica.com)

formed by binding two long polynucleotide strands. Each strand is a stack made up numerous nucleotides. A single nucleotide consists of a sugar, a phosphate, and one of four bases, adenine, cytosine, guanine, and thymine. The bases attached to both DNA strands pair with each other via weak, non-covalent bonds, known as hydrogen bonds. Adenine always interacts with thymine and cytosine interacts with guanine. Sugars and phosphates in both strands form the backbone on the outside of the double helix and while the bases (flat molecules) are facing inward forming the rings of the ladder. Because of the base complementary, the information contained within a double-stranded DNA molecule is two-fold redundant and can be easily replicated by separation of the strands and de novo synthesis (See figure 1.3). One end of each strand of DNA is called the $5'$ end and the other end is called the $3'$ end. The double helix structure is said to be antiparallel, that is, both strands run in opposite directions (one strand runs $5' \rightarrow 3'$ and the complementary strand runs $3' \rightarrow 5'$). One strand is called the template strand ($3' \rightarrow 5'$) and the other strand is called the non-template strand ($5' \rightarrow 3'$). It is the non-template strand that gets copied into an

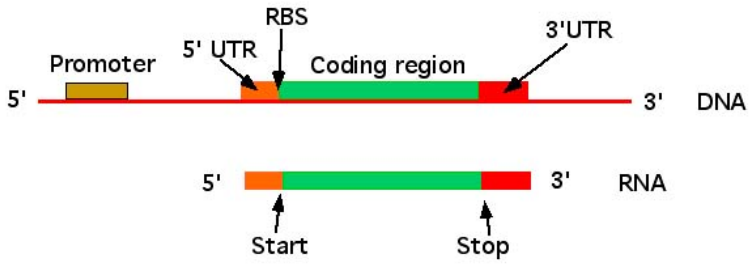


Figure 1.4: Basic structure of gene (http://nitro.biosci.arizona.edu)

mRNA molecule in the process of transcription (Figure 1.5. Not all the genes are expressed in every cell.

Each gene is a discrete segment of DNA, which is defined by being transcribed into a discrete short mRNA molecule, which in turn is translated into a discrete protein molecule. There are about 30,000 genes in human cells. Not all genes encode proteins. Some genes contain information to produce RNAs. A sequence located upstream ($5'$) of the protein coding region is called the gene's promoter (see figure 1.4 for the basic structure of gene). The promoter plays an important role in gene regulation when and where the gene is transcribed. Gene expression of DNA into protein proceeds via an important intermediate ribonucleic acid (RNA).

RNA is a long chain of nucleotides like DNA. In fact, the structure of RNA is very similar to that of DNA except for a few important details. RNA molecule is usually a single polynucleotide strand. RNA nucleotide contains four bases, like DNA, but thymine is replaced by uracil, which have pairs with adenine. RNA is transcribed from DNA by RNA polymerases (enzymes). There are many types of RNAs. Here, we introduce three RNAs, messengerRNA (mRNA), ribosomalRNA (rRNA), and transferRNA (tRNA) which are involved in protein synthesis.

Transcription of protein coding genes produces mRNA. mRNA is a "blueprint" (or "template") of a protein product (Figure 1.7). The bases in the mRNA are complementary to the template strand ($3' \rightarrow 5'$) of DNA. A group of three consecutive bases in the mRNA

template is called a codon.Each codon codes for a specific amino acid depending on the sequence of bases in each codon (see Figure 1.9). One codon specifies the start of the protein coding region on the mRNA. Three of codons serve as stop signals to terminate the translation process.

In order to translate the tripled code at the RNA level into a sequence of amino acids, a set of adopter molecules is needed. Another RNA produced by the transcription from DNA is a transfer RNA (tRNA). tRNA is a small molecule that delivers amino acids during the protein synthesis. One end of each tRNA contains an anticodon that is complementary to mRNA, so that tRNAs bind to mRNA during the translation. The other end attaches an amino acid. Depending on a codon, tRNA brings a different amino acid to mRNA to produce the amino acids sequence of a polypeptide. Figure 1.8 shows an example of an anticodon and an amino acid in tRNA with mRNA from Figure 1.7. tRNA with the anticodon "GAU" binds to the codon "CUA" . The corresponding amino acid to the codon "CUA" in the tRNA is "Leu" found in Figure 1.9. Cells contain one or more tRNAs for each of the 20 amino acid used during protein synthesis.

The last transcribed RNA is ribosomal RNA (rRNA) which makes up about 80% of RNA in the cell. rRNA is associated with specific proteins to form the ribosome. The ribosome is the central component of the protein production machinery in a cell. Each ribosome consists of two subunits called the large and small subunits. Two subunits of the ribosomes are attached to mRNA that stays in between them. Each ribosome accepts two tRNAs at a time. The ribosome functions as an enzyme that sticks together the amino acids that are delivered by the series of tRNAs.

In summary, gene expression is a process by which a gene's information is decoded into a cell's product, protein. It is a two-step process that begins with decoding of the genetic information from DNA to RNA (transcription) followed by translation of RNA into protein (translation).

Transcription and translation in prokaryotes and eukaryotes are a bit different in details and involving factors. We describe two processes which occur in eukaryotes below.

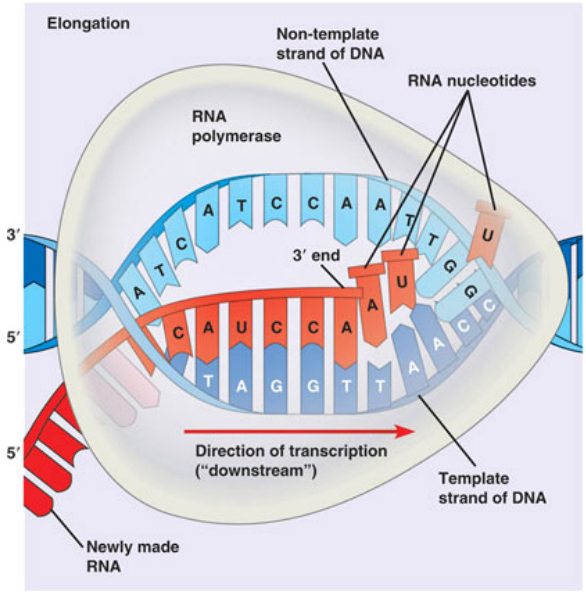


Figure 1.5: Elongation stage in translation (http://www.bio.miami.edu)

Transcription is the first step of gene expression and produces mRNA from the DNA template. Three phases occur during transcription: initiation, elongation, and termination. Transcription is initiated by transcription factors binding near the the gene promoter, which is called TATA (i.e. alternating thymine-adenine bonds) box . These factors prepare the DNA to bind RNA polymerase for a successful transcription. During the elongation (Figure 1.5), RNA polymerase (enzyme) moves along the DNA by unwinding a small portion of double helix in the $5' \rightarrow 3'$ direction of the template DNA strand. mRNA is produced by stacking of bonds that are complementary of the template DNA strand. Most factors are removed after initiation phase. Transcription is terminated at specific points after coding sequence. This area contains a sequence to stop transcription. Then, RNA polymerase is also released.

As we mentioned above, cells do not express all genes at once. Only about 15% of the human genome is expressed in any given cell, and the rest of the genes are inactive. In multicellular organisms, the subset of expressed genes varies depending on the types of cells. For example, a type of blood cell, the lymphocyte, produces antibodies by expressing genes that encode antibody polypeptide. The characteristics and roles of the cells are determined

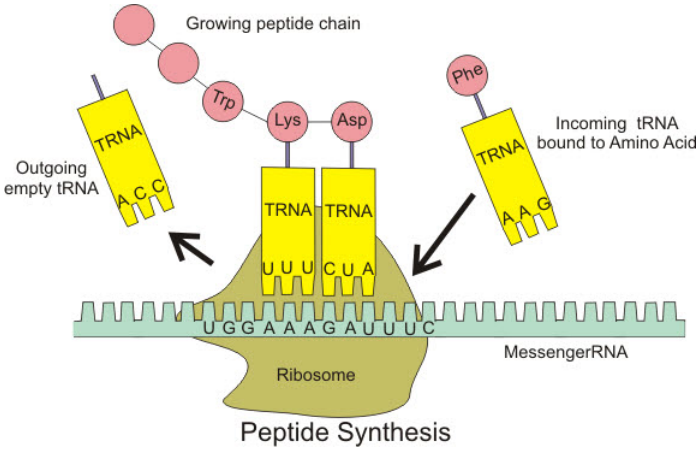


Figure 1.6: Elongation stage in translation (http://upload.wikimedia.org)

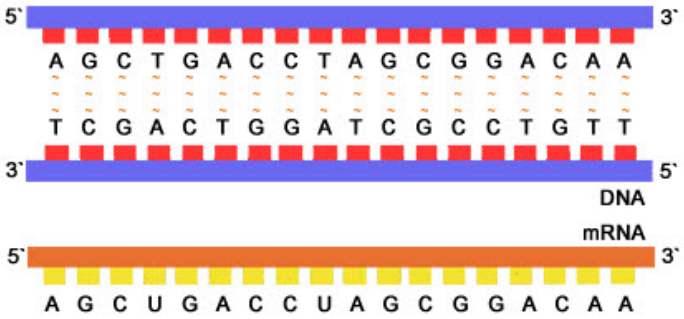


Figure 1.7: Bases in mRNA (http://library.thinkquest.org)

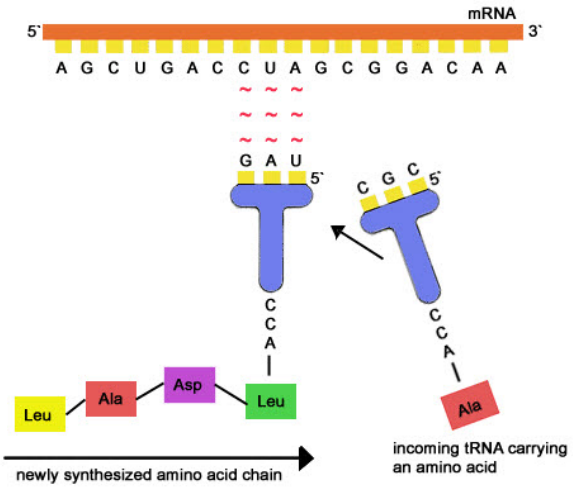


Figure 1.8: Anticodon and peptide in tRNA (http://library.thinkquest.org)

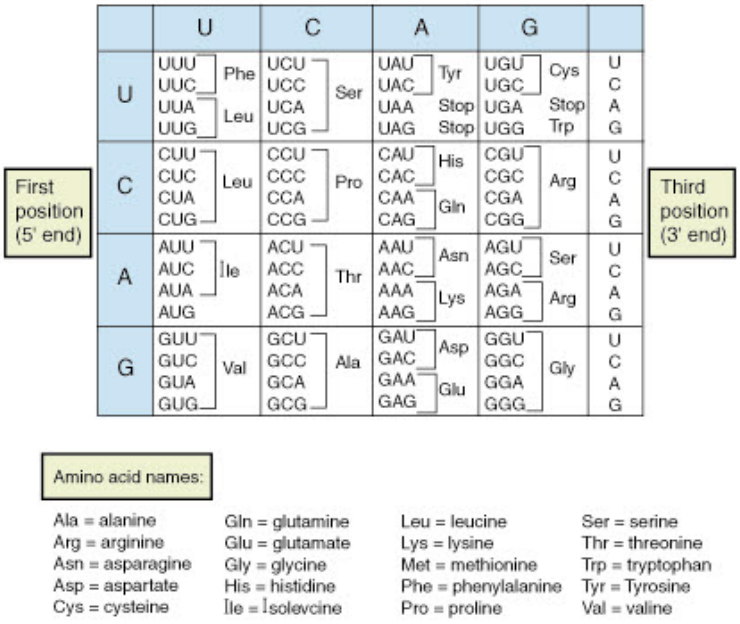


Figure 1.9: Genetic Code (http://img.sparknotes.com)

by its active gene set. The amount of proteins that a cell produces depends on the tissue, the developmental stage of the organism and the metabolic or physiologic state of the cell. The genes are under the control of complex patterns of regulation, so that the expressed genes are those that are necessary for the cell's functions. The pattern of gene expression can change during the lifetime of the cell, and an abnormal gene expression pattern can lead to the development of diseases. It is very important to understand the gene regulation to reveal the disease mechanism. Unfortunately, how genes are regulated is not yet fully understood. Transcriptional regulation is the best-studied form of regulation. Besides the regulation of transcription, expression of a gene may be controlled during RNA processing and transport, RNA translation, and by the post-translational modification of proteins. The degradation of mRNA and intermediate RNA products can also be regulated.

1.1.3 Technology

DNA microarray technology has greatly accelerated the pace of discovery in genetics. A DNA microarray is in the form of a large matrix whose size is thousands by thousands. In each spot, a single stranded DNAs with a gene-specific sequence, known as a probe, is attached [34]. Each gene in the genome is represented by one or, usually, multiple probes. In a microarray based gene separation experiment, first, mRNAs are extracted from sample cells that are examined. Second, complementary DNAs (cDNA), known as targets, are generated via reverse transcription using the extracted mRNA as template. Next, cDNA product is labeled using a fluorescence dye. The labeled cDNAs are deposited onto the surface of the microarray. cDNAs only bind to those probes on microarray that contain complementary bases. The binding, however, can not be seen by the human at this step. To see that, the hybridized microarray is scanned by fluorescence microscopy. Finally, the intensity of the target at each spot is analyzed using image analysis software. The intensity of fluorescent signal at each spot represents the amount of mRNA in the original sample of cells [34].

In order to use DNA microarray data for the currently proposed gene regulatory network identification problems, they need to be pre-processed, since microarray data almost certainly contains noise. Noise aries from different sources during the DNA microarray experiment such as incomplete extraction of the mRNA from the tissue, conversion to cDNA, hybridization, and so on. In this dissertation, we introduce two types of models, deterministic differential equation models and stochastic differential models. Deterministic differential models require pre-processed microarray data. Stochastic differential models can take raw microarray data.

1.2 Systems biology

In the late 1990s, scientists (biologists, mathematicians and engineers) began studying the hidden biological dynamics, such as interactions between genes and proteins, in a way that

has become known as systems biology. Systems biology has been one of the featured fields in modern science.

With recent advances in high-throughput technologies such as DNA microarray, it is possible to measure the spatial-temporal expression levels of thousands of genes at the system level. Data thus collected provides valuable descriptions of gene activities under various biochemical and physiological circumstances and allow one to identify their interactions at the system level, or to "reverse-engineer" the gene regulatory networks (GRNs), that is, to infer the underlying network structures from gene expression profiling. Understanding gene regulatory networks at the system level is a fundamental issue in the post-genomic era [10, 3, 15, 40, 57, 62, 80]. The interactions between genes for the purpose of gene regulation, development, discover etc. have been studied for many years and even decades J. Monod, M. Ptashne, C. N*ü*sslein Volhard, H. Varmus, ect. what distinguishes "systems biology" from those prior approaches is the goal of considering or incorporating all possible data at once, in a "holistic" fashion. Previously, interactions were usually reduced to simple patterns of one gene regulating on other gene, in the tried-and tested tradition of "reductionist" science. There are three types of interactions among genes within a GRN, namely, *activation, inhibition, and non-interaction*. The activation and inhibition are represented by interlocking positive and negative effects of one gene on another between genes, respectively. Because most GRNs of interest involve many constituents (such as genes, RNAs, proteins, and other molecules), which are connected and interact in a very complex fashion, an intuitive understanding of the network and its dynamics is difficult to achieve. Consequently, formal mathematical and statistical methods and computer tools for modeling and simulation of GRNs become indispensable. In the past fifteen years, various methods and approaches have been introduced to study gene regulatory network identification (and functional prediction). Among them are statistical methods such as Boolean, Bayesian, graph network, and neural network methods, as well as machine learning algorithms [78, 22, 39, 49, 54, 53, 61, 26].

Boolean network models make use of Boolean variables to infer GRNs. In a Boolean model, the state of a gene is described by a Boolean variable 0 (inactive) or 1 (active) and

interactions between the genes are expressed by Boolean functions which calculate the state of a gene based on the states of some other genes. In the Boolean network formalism, the mRNA expression levels are discretized to be active or inactive and intermediate expression levels are ignored. So the discretization may cause the loss of significant information, hence, give unrealistic GRNs. In Bayesian network models, the relationships between genes are expressed by graph structures whose vertices and directed edges represent variables (genes) and dependencies, respectively. Variables without connections are conditionally independent. They together form a direct acyclic graph which has no loops. Bayesian methods infer GRNs using direct acyclic graphs and conditional probability distributions and their statistical theories. They are not proper to infer GRNs which contain loops [18]. Moreover, Bayesian methods only infer GRNs' graphic structures, they do not provide the dynamical aspects of gene regulations.

Alternatively, inspired by the electrical engineering paradigm, some successful ideas such as systems and control theories from electrical circuitry have been borrowed to study genomic circuitry [87]. The defining feature of such an approach is that it relies on mathematical modeling and numerical/computer simulation. Since such mathematical models are often described by differential equations (DEs), we shall refer this approach as *DE-based approach or DE-modeling*. Recently, various DE models and methods have been developed in the literature [17, 19, 40, 28, 80, 62, 77, 86, 26].

1.3 Background and literature review of DE-based models of gene regulatory networks

1.3.1 Dynamical models of gene regulatory networks

In order to construct Gene Regulatory Networks (GRNs) identification models based on differential equations (DEs), the concentrations of mRNA, proteins, other molecules, which take values of nonnegative real numbers, are often used as the primary variables since they are measurable from the microarray data [48] and assumed to vary continuously in time.

DE based GRN identification models relate to an external perturbation which changes the rate of change of transcript concentrations in a cell. Let t be the time, n denote the number of genes in the GRNs, and $y_i(t)$ represent the gene transcript concentration of gene i at time t in the network, then for gene i, $y_i(t)$ is assumed to satisfy the following rate equations [26, 40, 80]:

$$\frac{dy_i(t)}{dt} = f_i(y_1(t), y_2(t), \cdots, y_n(t)) + b_i v_i \quad \text{for} \quad i = 1, 2, \cdots, n, \tag{1.3.1}$$

where $f_i : \mathbf{R}^n \to \mathbf{R}$ is a scalar valued influence function and a (small) constant quantity v_i is an external perturbation to each gene. b_i is the impact of the perturbation v_i on i^{th} gene. v_i could be a function of the time depending on what type of experimental data is collected. Hence, the entire GRN via gene expression data can be modeled by an n-dimensional dynamical system of rate equations:

$$\frac{d\mathbf{y}(t)}{dt} = \mathbf{F}(\mathbf{y}(t)) + \mathbf{b} \cdot \mathbf{v}, \tag{1.3.2}$$

where $\mathbf{y}(t) = [y_1(t), y_2(t), \cdots, y_n(t)]^T$, $\mathbf{R}_+ := [0, \infty) \to \mathbf{R}_+^n$, $\mathbf{F}(\mathbf{y}) = [f_1(\mathbf{y}), f_2(\mathbf{y}), \cdots, f_n(\mathbf{y})]^T$, and $\mathbf{v} = [v_1, v_2, \cdots, v_n]^T$ (for the time dependent external perturbation $\mathbf{v}(t) = [v_1(t), v_2(t), \cdots, v_n(t)]$). Clearly, the choice of influence function $\mathbf{F}$ is the crucial first step to build good mathematical models of the form (1.3.2), and different choices of $\mathbf{F}$ may lead to different conclusions about the structure and the quantitative relations between genes in the GRN. Several choices of $\mathbf{F}$ such as linear and piecewise linear functions have been proposed and studied in the literature (see [26] and the references therein), however, their usefulness is severely hampered by the lack of *in vivo* or *in vitro* measurements of the (kinetic) parameters in the rate equations. In addition to the linear models, nonlinear influence functions should be considered, since in general, most of biological networks are governed by nonlinear dynamics. There, however, have been very few nonlinear models proposed in literature because of the difficulties in estimating parameters with the small number of data compared to the large number of parameters. Instead of constructing the

biologically well-defined nonlinear function $\mathbf{F}$, most of approaches for the DE modeling consider the first order approximation of the model (1.3.2) near a steady state solution, where gene expression values do not change significantly over time and their gene expression levels are measurable in a lab experiment [40, 41, 4]. That is, one seeks a solution of the form $\mathbf{y}(t) = \mathbf{y}_0 + \delta\mathbf{x}(t)$ for $|\delta| \ll 1$, where $\mathbf{y}_0$ is the steady state solution of (1.3.2). Plugging $\mathbf{y}(t)$ into (1.3.2), using Taylor formula and the fact that $\mathbf{F}(\mathbf{y}_0) \approx 0$, and neglecting the higher order terms in δ we get the following linearized differential equation model:

$$\frac{d\mathbf{x}(t)}{dt} = A\mathbf{x}(t) + P\mathbf{u}, \tag{1.3.3}$$

where $\mathbf{u} = \delta^{-\mathbf{1}}\mathbf{v}$ and $A := [a_{ij}]$ denotes the $n \times n$ Jacobian matrix of $\mathbf{F}$ at $\mathbf{y}_0$, i.e. $a_{ij} = \dfrac{\partial f_i}{\partial y_i}(\mathbf{y}_0)$. Each entry of A encodes the regulatory interaction (activation, inhibition, and non-interaction) between the genes in the network and its magnitude measures the strength of the interactions. If $a_{ij} > 0$, the j^{th} gene activates the i^{th} gene, which means that the rate of change in production of i^{th} is increasing. For $a_{ij} = 0$, it means that there is no interaction between the j^{th} and the i^{th} genes in the network. If $a_{ij} < 0$, the j^{th} gene inhibits the i^{th} gene, which means that the rate of change for the production of i^{th} gene is decreasing. The matrix A is often called the *influence matrix* in the biology literature [41]. The $n \times q$ matrix $P := [p_{il}]$ represents the effect of l perturbations on n genes. $p_{il} \neq 0$ means that the l^{th} perturbation has a direct effect on the i^{th} gene. Otherwise, there is no direct effect of the l^{th} perturbation on the i^{th} gene. In recent studies, the linearized DE modeling largely uses two different forms of data [3, 18]. One is time-series data and the other one is steady-state data. Depending on types of data, (1.3.3) is slightly different. In the case of time-series data, one uses (1.3.3). Depending on types of the external perturbation, the amount of external perturbations changes in time as gene expression level and is measurable. If $\mathbf{u}$ depends on time [4], then (1.3.3) becomes

$$\frac{d\mathbf{x}(t)}{dt} = A\mathbf{x}(t) + P\mathbf{u}(t). \tag{1.3.4}$$

Studies with steady-state data, mRNA concentrations and the external perturbations are collected at time points $\{t_i\}$, where $\frac{d\mathbf{x}(t_i)}{dt} = 0$, for each external perturbation. Hence, (1.3.3) can be simplified as

$$0 = A\mathbf{x} + P\mathbf{u}, \tag{1.3.5}$$

which is independent of time [40, 29, 55]. Although many studies use steady-state data, it has the limitation to reflect the dynamics of interactions in the network. Also, it requires more than one perturbation to make the problem feasible. The choice of perturbations is critical because it needs to reflect all the characteristics of genes in the network. As one can imagine, the amount of perturbations can play an important role. Too large perturbations result in gene expression level away from the original steady-state, and too small perturbations produce gene expression data that is not large enough to infer its structure. It is improper to apply the above approach to a gene regulatory network which does not have a steady state.

For both approaches using time-series and steady-state data, one replaces $P\mathbf{u}$ by $\mathbf{u}$ in (1.3.4) and (1.3.5). For time-series gene expression data, the external perturbations to each gene are given once at the beginning of an experiment and the level of perturbations at each time point remains the same as at the beginning. For steady-state data, it is required to know which gene has been directly perturbed in each perturbation experiment [40, 3].

In this dissertation, we adopt the linearized DE-modeling using time-series gene expression data and assuming $P = I$ (identity matrix), that is, we consider the following dynamic model:

$$\frac{d\mathbf{x}(t)}{dt} = A\mathbf{x}(t) + \mathbf{u}. \tag{1.3.6}$$

However, we like to point out that our approach also applies to the more general model (1.3.4) with a minor modification, see Section 2.4 for details. The problem is now reduced to the one that identifies the entries of the influence matrix A using only gene expression data, $\mathbf{x}(t_j) = [x_1(t_j), x_2(t_j), \cdots, x_n(t)]^T$ for $j = 1, 2, \cdots, m$. If $m \geq n - 1$, i.e., the number of time points is no less than the number of species in the network, then the above problem

is *overdetermined* and its solution can be computed easily using the least squares method. However, for a large network ($100 < n < 30,000$) it is impractical to measure gene expressions at n time points. Typically, $2 < m < 100$. Thus, practically the above problem is *underdetermined* and hence expected to have infinitely many solutions. We note that the above problem is often referred as the *dimensionality problem* in the literature (see [26] and the references therein).

1.3.2 Discretization

The gene expression data $\mathbf{x}(t)$ is measured at $m+1$ time points $\{t_j\}_{j=1}^{m+1}$ in a lab experiment. Therefore, we can approximate $\dfrac{d\mathbf{x}(t)}{dt}$ at each time point t_j $(1 \leq j \leq m)$ by

$$\frac{d\mathbf{x}(t_j)}{dt} \approx \frac{\mathbf{x}(t_{j+1}) - \mathbf{x}(t_j)}{t_{j+1} - t_j}. \tag{1.3.7}$$

Furthermore, the discretized model can be phrased as the following matrix equation problem (P): Given $X, Y \in \mathbf{R}^{n\times m}$, $n >> m > 1$, find $A \in \mathbf{R}^{n\times n}$ such that

$$AX = Y. \tag{1.3.8}$$

In equation (1.3.8), the columns of the matrix X represents a time-series of the concentration $\mathbf{x}(t_j)$ measured at m time incidences $\{t_j\}$, columns of the matrix Y stands for discrete approximations of the rate $\left(\dfrac{d\mathbf{x}}{dt} - \mathbf{u}\right)$ at the same m time incidences.

A nonzero value at the (i, j) entry of the matrix A indicates that there is a regulatory interaction between the i^{th} gene and the j^{th} gene, the magnitude of the value represents the strength of the interaction. A positive and negative entry of A represent activation and inhibition, respectively.

1.4 Scope and contributions of the dissertation

An important and challenging problem in post-genomic research is to infer GRN from gene expression profiling experiments. Unfortunately, the huge number of constituents and their complex relationships in the cell make the mathematical modeling and numerical simulations of large-scale biological networks very challenging. From the mathematical point of view, one of the main obstacles is that the amount and the quality of the experimental data at hand is often insufficient for an unequivocal assignment of the model parameters. Mathematically, that means *the problem of reverse engineering gene regulatory networks is severely ill-posed*. This may sound like a paradox because the naive perception is that high-throughput technologies produce too much data to be analyzed. The perception is indeed true from the static point of view, however, it is not the case from the dynamic point of view. Due to economical and practical reasons, it is too expensive to measure gene expression profiles for thousands of genes at a large number (≥ 200) of time points.

The restriction $n >> m > 1$ in problem (P) is due to the above reasons. Consequently, system (1.3.8) is strongly *underdetermined*. Hence, problem (P) is mathematically *ill-posed* because of the non-uniqueness of solutions. To see this, suppose A is a solution of equation (1.3.8), let $\mathbf{z} \in \mathbf{R}^n$ be any nonzero vector and $\mathbf{z}^T$ is orthogonal to all columns of X (i.e., $\mathbf{z}^T$ belongs to the null space of the matrix X), then it is trivial to check that the matrix $A + \mathbf{z}\mathbf{z}^T$ is also a solution of equation (1.3.8).

The most popular and widely used method for selecting a solution is the *least squares method*, which seeks A as a minimizer of the following minimization problem:

$$\min_{B \in \mathbf{R}^{n \times n}} \|Y - BX\|_F^2, \tag{1.4.1}$$

where $\|\cdot\|_F$ denotes the Frobenius matrix norm, that is,

$$\|X\|_F := \sum_{i=1}^{n} \sum_{j=1}^{m} |x_{ij}|^2. \tag{1.4.2}$$

It is easy to check that A is a solution to (1.4.1) if and only if A satisfies the following *normal system*

$$AXX^T = YX^T \qquad \text{or} \qquad XX^TA^T = XY^T. \tag{1.4.3}$$

Unfortunately, since $n > m$, the matrix $XX^T \in \mathbf{R}^{n\times n}$ is a *singular* matrix, hence, solutions of (1.4.3) are *not unique*! Consequently, the least squares method fails to solve equation (1.3.8) unless some additional remedies are taken.

The very first goal of this dissertation is to propose a new approach to identify the biologically meaningful solution from the set of infinitely many possible solutions. In Chapter 2, we introduce our new approach for selecting the unique biological solution to problem (P). In our approach, we present a general variational framework for solving the ill-posed problem (P). In our framework, equation (1.3.8) is treated as a constraint, we then define the desired biological solution as the unique solution of some constrained variational problem with constraint (1.3.8). Furthermore, it is important to a choose proper energy functional in the constrained variational problem so that the resulting problem is well-posed. Thus, for the DE-modeling to be successful, the advances must be made at the level of developing more accurate and realistic models and fast and efficient computational methods for computing the solutions of these models. Various examples of energy functionals will also be presented in Chapter 2.

In each of Chapter 3-6, we analyze each of the proposed models in detail. This includes to examine the well-posedness of the model, to design and implement efficient numerical methods and algorithms to compute the solution. We also test and validate each of our models using real world *S. cerevisiae* (yeast) and *E. coli* microarray data.

Because of uncertainties, such as measurement errors, equipment malfunction, and insufficient information, scientific data such as those obtained from microarray technologies are typically noisy. The second goal of this dissertation is to develop stochastic DE models for gene regulatory networks by considering and incorporating those uncertainties into mathematical models. The new models require the use of stochastic and statistic tools and

methods. In Chapter 7, we introduce a new stochastic differential equation model in place of (1.3.6) and design and test numerical methods and algorithms to compute the solutions of the new models.

Another issue in reverse engineering of GRN is post-processing. Based on previous studies about the biochemical networks, cellular networks are usually not fully connected and have special structures. Hence, the influence matrix A is expected to be sparse (i.e., most of its entries are zeros). Most approaches for the linear DE-modeling assume the loose connectivity of inferring networks. However, the inferred network matrices A from other approaches including our approach are often dense although most entries of A are expected to be very small, that is, the networks are fully connected. So we are forced to make a delicate but important decision, that is, to decide which small entries in an inferred influence matrix A should be set to zero and which should be kept. In the literature (cf. [87]), the threshold values are usually decided in some ad hoc or arbitrary manner, and often lead to less satisfactory or unsatisfactory results. The final main goal of this dissertation is to develop an automatic thresholding technique. For the deterministic DE-modeling, we choose a threshold value manually. For the stochastic DE-modeling, we propose a thresholding method based on the Random Matrix Theory (RMT). The random matrix theory was recently introduced and developed for automatically post-processing biological networks [63, 64]. Clearly, it is a non-trivial and non-intuitive method. In Chapter 7, we introduce the RMT and present our investigation on the RMT applied to our variational framework as the default method for post-processing the identified influence matrix A.

The dissertation is concluded by a brief summary and a list of future directions and works in Chapter 8.

Chapter 2

Variational Methods for Gene Regulatory Network Identification Based on Differential Equation Modeling

In this chapter, we first define a biological solution for problem (P) based on a constrained variational framework. We then establish the existence and uniqueness of biological solutions for the constrained variational problem under some structure assumptions on the energy functional. Examples of the energy functional to be studied in this dissertation are also introduced. We also present an equivalent unconstrained variational formation for the constrained variation problem using the Singular Value Decomposition (SVD) theory. Finally, we show how our variational framework can be adopted to cover the general model (1.3.4).

2.1 The general framework of variational methods

We define a biological solution to problem (P) as follows:

Definition 2.1.1. Let $\mathcal{L} : \mathbf{R}^{n\times n} \longrightarrow \mathbf{R}_+ := [0, \infty)$ be a pre-determined (energy) functional on $\mathbf{R}^{n\times n}$, we call $A \in \mathbf{R}^{n\times n}$ a *biological solution* to problem (P) if A is a minimizer of the following variational problem:

$$\min_{B\in\mathbf{R}^{n\times n}} \mathcal{L}(B) \quad \text{subject to} \quad BX = Y. \tag{2.1.1}$$

In other words, the variational problem (2.1.1) is our selection criterion for picking up the biologically relevant solution among infinitely many possible choices.

Clearly, the most important component of the general framework is to design the energy functional $\mathcal{L}$. Mathematically, it is not hard to construct an energy functional such that the minimization problem (2.1.1) has a unique solution. In fact, any *strictly convex* functional will work. On the other hand, it is an art (and also a science) to design and construct a functional $\mathcal{L}$ which will give a good "biological solution". This requires biological intuition, knowledge, experience, and insights, in addition to mathematical sophistication.

The following two theorems show existence and uniqueness of solutions to problem (P).

Theorem 2.1.1. *(Existence)* Suppose $\mathcal{S} := \{B \in \mathbf{R}^{n\times n}; BX = Y\}$ be the solution space of the constraint equation to problem (P) and that $\mathcal{L}$ is a lower semicontinuous functional and it is equivalent to a matrix norm $\|\cdot\|$ on $\mathcal{S}$, that is, there exists c_0, $c_1 > 0$ such that $c_0\|B\| \le \mathcal{L}(B) \le c_1\|B\|$, $\forall B \in \mathcal{S}$. Then there exists a biological solution to problem (P).

Proof. Suppose

$$m := \inf_{B\in S} \mathcal{L}(A) < \infty.$$

Let $\{A_n\}_{n=1}^{\infty} \in S$ be a minimizing sequence for $\mathcal{L}$, that is, $\mathcal{L}(A_n) \to m$ as $n \to \infty$. Since $\mathcal{L}(\cdot)$ is bounded by a matrix norm for all matrices in the solution space, then $\{A_n\}_{n=1}^{\infty}$ is a bounded sequence. Hence, the finite dimensionality of $\mathcal{S}$ implies that there exists a convergent subsequence, $\{A_{n_k}\}_{k=1}^{\infty}$, such that $\lim_{k\to\infty} A_{n_k} = A$.

By the fact that the functional $\mathcal{L}$ is lower semicontinuous, we get

$$m \leq \mathcal{L}(A) \leq \liminf_{k\to\infty} \mathcal{L}(A_{n_k}) = m.$$

Hence, $\mathcal{L}(A) = m$. That is, A is a solution of problem (2.1.1). □

Theorem 2.1.2. *(Uniqueness 1)* Suppose $\mathcal{L}$ is a strictly convex functional, then the biological solutions to problem (P) is unique.

Proof. Suppose that A_1 and A_2 are two distinct solutions to (2.1.1). Since $\mathcal{L}$ is strictly convex (cf. Definition A.0.1), then for every $\alpha \in (0, 1)$ we have

$$\mathcal{L}\big(\alpha A_1 + (1-\alpha)A_2\big) < \alpha\mathcal{L}(A_1) + (1-\alpha)\mathcal{L}(A_2) = \mathcal{L}(A_1) = \mathcal{L}(A_2),$$

which contradicts with the assumption that A_1 and A_2 are solutions to (2.1.1). Hence, the minimization problem (2.1.1) has a unique solution when the functional $\mathcal{L}$ is strictly convex. □

The most widely used functionals of matrices are matrix norms. For a choice of matrix norm as $\mathcal{L}$, it is easy to show that such an $\mathcal{L}$ is not strictly convex in the sense of Definition A.0.1, so it is necessary to verify the uniqueness of the biological solutions of problem (P) using Definition A.0.2 or Definition A.0.3 (see Appendix A). Since they are equivalent, it is enough to show uniqueness with Definition A.0.2.

Theorem 2.1.3. *(Uniqueness 2)* Suppose $\mathcal{L} = \|\cdot\|$ represents a strictly convex matrix norm in the sense of Definition A.0.2, then the biological solutions to problem (P) is unique.

Proof. Assume that A_1 and A_2 are two solutions to (2.1.1). Then, $\|A_1\| = \|A_2\|$, and

$$\begin{aligned} \|A_1 + A_2\| &\leq \|A_1\| + \|A_2\| = 2\|A_1\|, \\ \left\| \frac{A_1 + A_2}{2} \right\| &\leq \|A_1\|. \end{aligned}$$

Since $\|A_1\|$ is a solution of (2.1.1), we must have

$$\left\| \frac{A_1 + A_2}{2} \right\| = \|A_1\|.$$

Therefore, by the strict convexity assumption we get $A_1 = A_2$. Hence, the uniqueness holds. □

In the next section, we introduce several examples of the energy functional which will be analyzed in later chapters in detail.

2.2 Examples of energy functional $\mathcal{L}$

We emphasize that different choices of strictly convex functional $\mathcal{L}$ will give different models, and hence, result in different solutions to problem (P), although these solutions are expected and hoped to be qualitatively similar. The viability and applicability of a model might be problem-specific, and need to be tested carefully on benchmark problems. In the following we give a few examples of the energy functional $\mathcal{L}$. Detailed analysis and validations of these models will be presented in the later chapters.

1. The least squares model (LSM)

The first and obvious choice of $\mathcal{L}$ is

$$\mathcal{L}(B) = \| BX - Y \|_F^2. \tag{2.2.1}$$

With this choice of $\mathcal{L}$ problem (2.1.1) reduces to the least squares problem (1.4.1). Hence, we recover the least squares model. As noted early in Section 1.3.2, the least squares model is ill-posed since it has multiple solutions in the case $n > m$. The deep mathematical reason for the non-uniqueness is that the above functional $\mathcal{L}$ is convex but *not* strictly convex.

2. The average minimum strength model (AMSM)

Next example of $\mathcal{L}$ is

$$\mathcal{L}(B) = \| B \|_F := \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |b_{ij}|^2}, \tag{2.2.2}$$

that is, $\mathcal{L}(B)$ is defined as the matrix Frobenius norm of B.

3. The c_p minimum strength model (c_pMSM)

Next example of the energy functional $\mathcal{L}$ is the matrix c_p norm. For $1 \leq p \leq \infty$, let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ be singular values of $B \in \mathbf{R}^{n \times n}$ (cf. Appendix A). Define

$$\mathcal{L}(B) = \| B \|_{c_p} := (\sum_{i=1}^{n} \sigma_i^p)^{\frac{1}{p}}. \tag{2.2.3}$$

One easily notices that (2.2.3) is the same model as (2.2.2) for $p = 2$. For $p = \infty$ and $p = 1$, (2.2.3) is equivalent to the matrix 2-norm ($\|B\|_2 = \sigma_1$) and the trace (nuclear) norm, respectively.

4. The column minimum strength model (CMSM)

CMSM uses the matrix L^1-norm as the functional.

$$\mathcal{L}(B) = \| B \|_{L^1} := \max_{1 \leq j \leq n} \sum_{i=1}^{n} |b_{ij}|. \tag{2.2.4}$$

5. The row minimum strength model (RMSM)

Next example of the functional is the matrix L^∞-norm.

$$\mathcal{L}(B) = \| B \|_{L^\infty} := \max_{1 \leq i \leq n} \sum_{j=1}^{n} |b_{ij}|. \tag{2.2.5}$$

6. The L^p minimum strength model (L^pMSM)

Another family of the energy functional $\mathcal{L}$ is using the matrix L^p norm,

$$\mathcal{L}(B) = \|B\|_{L^p}^p, \tag{2.2.6}$$

where

$$\| B \|_{L^p} := \sup_{\mathbf{x}\in\mathbf{R}^n, \mathbf{x}\neq 0} \frac{\|B\mathbf{x}\|_p}{\|\mathbf{x}\|_p}, \quad 1 < p < \infty,$$

and $\|B\mathbf{x}\|_p$ and $\|\mathbf{x}\|_p$ denote the vector p-norm of $B\mathbf{x}$ and $\mathbf{x}$, respectively.

7. The entrywise minimum strength model (EMSM)

For $1 \leq p \leq \infty$, the entrywise matrix p-norm is

$$\| B \|_{e_p} := \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |b_{ij}|^p \right)^{\frac{1}{p}}.$$

For $p = \infty$, the entrywise matrix p-norm is same as the max norm

$$\| B \|_{e_\infty} := \max(|b_{ij}|).$$

For the choice of the entrywise matrix p-norm, we define the energy functional $\mathcal{L}$ as

$$\mathcal{L}(B) = \|B\|_{e_p}^p, \quad 1 \leq p \leq \infty. \tag{2.2.7}$$

Unfortunately, many of above models do not immediately promise a unique solution since these norms are not strictly convex. Strictly convex norms are (2.2.2), (2.2.3)($1 < p < \infty$), (2.2.6), and (2.2.7). Although the others are not strictly convex, we still expect a unique minimizer to (2.1.1) because of the constraint. Note that strict convexity is only a sufficient condition for the uniqueness of minimizers.

We shall study numerical algorithms for different choices of $\mathcal{L}$ including AMSM, c_pMSM,

CMSM, RMSM, L^pMSM, and EMSM in the later chapters. We propose and implement numerical algorithms on both synthetic mircroarray data and real microarray data.

2.3 Derivation of unconstrained variational problem

Once the functional $\mathcal{L}$ is determined, the next question is how to convert the constrained variational problem to an unconstrained problem. The general approach to this matter is to use the Lagrange multiplier method. That is, there exists a unique constant $\lambda > 0$, called the Lagrange multiplier, such that the constrained variational problem (2.1.1) is equivalent to the following unconstrained variational problem

$$\min_{B \in \mathbf{R}^{n \times n}} \mathcal{L}_\lambda(B),$$

where

$$\mathcal{L}_\lambda(B) := \mathcal{L}(B) + \lambda \|BX - Y\|_F^2.$$

To compute the solution to the above unconstrained problem, it is crucial to find the right λ, which is often difficult to compute.

We introduce a different approach to get rid of the constraint. We first decompose the biological solution matrix B as $B = B_1 - B_0$, where B_0 is a general solution of the homogeneous equation $B_0 X = 0$ and B_1 is a particular solution of $B_1 X = Y$. As mentioned in Section 1.3.2, there are infinitely many candidates for choosing a particular solution B_1 since it is the ill-posed problem. To compute the general solution B_0, we use the Singular Value Decomposition (SVD) (see Appendix A) which plays an important role in our approach .

We first transpose the equation $B_0 X = 0$ and let $X^T = U \Lambda V^T$ be a SVD of X^T, where

$$U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mm} \end{pmatrix} \in \mathbf{R}^{m \times m},$$

$$V = \begin{pmatrix} v_{11} & u_{12} & \cdots & v_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{pmatrix} \in \mathbf{R}^{n \times n}$$

are orthogonal matrices, Λ is diagonal matrix

$$\Lambda = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & 0 \\ 0 & 0 & \cdots & \sigma_m & \cdots & 0 \end{pmatrix} \in \mathbf{R}^{m \times n}.$$

Then, $X^T B_0^T = U \Lambda V^T B_0^T = 0$. Hence, column vectors of B_0^T (or row vectors of B_0) must belong to the null space of X^T. Thus, let $\mathbf{b}_k$ denote the k^{th} column vector of B_0^T, we have $\mathbf{b}_k = c_{kr+1}\mathbf{v}_{r+1} + c_{kr+2}\mathbf{v}_{r+2} + \cdots + c_{kn}\mathbf{v}_n$, $k = 1, 2, \cdots, n$ and r is rank of X^T. Consequently, B_0^T has the following decomposition:

$$\begin{aligned} B_0^T &= \begin{pmatrix} \mathbf{v}_1, & \mathbf{v}_2, & \cdots, & \mathbf{v}_n \end{pmatrix} \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ c_{1r+1} & c_{2r+1} & \cdots & c_{nr+1} \\ \vdots & \vdots & \vdots & \vdots \\ c_{1n} & c_{2n} & \cdots & c_{nn} \end{pmatrix} \\ &= VC^T. \end{aligned}$$

Equivalently, $B_0 = CV^T$ where

$$C = \begin{pmatrix} 0 & \cdots & 0 & c_{1r+1} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & c_{nr+1} & \cdots & c_{nn} \end{pmatrix} = (\mathbf{0} \;\; C_0),$$

0 stands for $n \times r$ zero matrix, and $C_0 \in \mathbf{R}^{n\times(n-r)}$. By the decomposition of the biological solution matrix, we get

$$B = B_1 - C_0 V_0^T,$$

where

$$V_0 = [\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \cdots, \mathbf{v}_n].$$

Substituting the above B into (2.1.1), we then convert the constrained minimization problem into the following unconstrained minimization problem:

$$\min_{C_0 \in \mathbf{R}^{n\times(n-r)}} \mathcal{L}(B_1 - C_0 V_0^T). \tag{2.3.1}$$

It is easy to see that the number of unknowns in (2.3.1) is $n(n-r)$ instead of n^2.

Regarding to the choice of B_1, we shall prove in Chapter 3 that the solution of the AMSM defined by (2.1.1) and (2.2.2) will provide a natural and convenient choice.

2.4 Extension to the general model (1.3.4)

Some researchers use the equation

$$\frac{d\mathbf{x}(t)}{dt} = A\mathbf{x}(t) + P\mathbf{u}(t) \tag{2.4.1}$$

to infer GRN and reveal the effects of external perturbations on genes using time-series data. Gene expression $\mathbf{x}(t)$ and the external perturbations $\mathbf{u}(t)$ for $j = 1, 2, \cdots, m$ can be measured in a lab. We note that the number of external perturbations need not to be same as the number of genes in the network.

$\frac{d\mathbf{x}(t)}{dt}$ is approximated as before. Then the discretized model of (2.4.1) can be written in matrix form: Given $X, W \in \mathbf{R}^{n\times m}$ and $U \in \mathbf{R}^{q\times m}$, find $A \in \mathbf{R}^{n\times m}$ and $P \in \mathbf{R}^{n\times q}$ such that

$$W = AX + PU. \tag{2.4.2}$$

The columns of the matrix X represents a time-series of the concentration $\mathbf{x}(t_j)$ measured at m time incidences $\{t_j\}$, column of matrix W stands for discrete approximations of the rate $\dfrac{d\mathbf{x}(t)}{dt}$ at the same m incidences $\{t_j\}$, and the column of U represents a time-series of the external perturbations measured at the same m incidences $\{t_j\}$. The entries A indicates regulatory interactions between genes in the network. A nonzero element at the (i, l) of matrix P represents that i^{th} gene is a direct target of the l^{th} perturbation. Problem (2.4.2) is ill-posed since the number of time points we can have at hand is usually much smaller than the number of genes and external perturbations. Hence, there are infinitely many solutions. To handle this problem, Bansal *et al.*[4] used the method of interpolation. They increased the number of time points by using a cubic smoothing spline filter and piecewise cubic spline interpolation.

Our variational approach can be applied to this problem by rearranging the matrices in (2.4.2). We can phrase the equation (2.4.2) as the following matrix equation problem:

$$W = \begin{pmatrix} A_{n\times n}, P_{n\times q} \end{pmatrix} \begin{pmatrix} X_{n\times m} \\ U_{q\times m} \end{pmatrix}.$$

Hence, we have a new form of problem: Given $W \in \mathbf{R}^{n\times m}$ and $X_U \in \mathbf{R}^{(n+q)\times m}$, find $A_P \in \mathbf{R}^{n\times(n+q)}$ such that

$$W = A_B X_U, \tag{2.4.3}$$

where

$$A_P = \begin{pmatrix} A_{n\times n}, P_{n\times q} \end{pmatrix} \quad \text{and} \quad X_U = \begin{pmatrix} X_{n\times m} \\ U_{q\times m} \end{pmatrix}.$$

Finally, we can apply our variational approach to problem (2.4.3) as described in Sections 2.1-2.3.

Chapter 3

The Average Minimum Strength and the c_p Minimum Strength Models

The Average Minimum Strength Model (AMSM) uses the functional $\mathcal{L}(\cdot) = \|\cdot\|_F$, the Frobenius matrix norm. For this choice the abstract variation model (2.1.1) becomes

$$\min_{B\in\mathbf{R}^{n\times n}} \|B\|_F \quad \text{subject to} \quad BX = Y. \tag{3.0.1}$$

From the definition of $\|\cdot\|_F$, it is clear that the AMSM treats every entry of the network matrix equally, At the minimizer, the "energy" of the network is minimized in a square averaged sense. To some extent, this is one way to describe the "small-world" property of GRN (i.e. each gene in the GRN interacts with a few genes). We remark that the AMSM can be easily modified to incorporate the anisotropy (if a priori information is known) of some GRNs by defining $\mathcal{L}$ to be a weighted Frobenius norm functional.

The c_p Minimum Strength Model (c_pMSM) uses the functional $\mathcal{L}(\cdot) = \|\cdot\|_{c_p}$, the c_p

matrix norm (or the schatten norm). Then, the abstract variation model (2.1.1) becomes

$$\min_{B\in\mathbf{R}^{n\times n}} \|B\|_{c_p} \quad \text{subject to} \quad BX = Y. \tag{3.0.2}$$

To construct an numerical algorithm to approximate the minimizer of (3.0.2), we solve the unconstrained minimization problem which was introduced in Chapter 2. The matrix c_p-norm has an important unitary invariance property (cf. Appendix A). By this property, we have an interesting result about the solution of (3.0.2).

The goals of this chapter include establishing the well-posedness of (3.0.1), designing efficient numerical algorithms to compute the solution of (3.0.1), and doing numerical experiments to check the performance of the model and the algorithms. Also, we prove the general result about the solution of (3.0.2).

3.1 The average minimum strength model

3.1.1 Existence and uniqueness of minimizers

For the AMSM, there holds the following nice characterization result in the case when X has full rank, which is often satisfied by microarray data.

Theorem 3.1.1. Suppose that the matrix X has full rank. Then the unique solution to problem (3.0.1) is given by

$$A = Y(X^TX)^{-T}X^T. \tag{3.1.1}$$

Proof. First, since X has full rank, so does (X^TX), hence $(X^TX)^{-1}$ exists. Therefore, $A = Y(X^TX)^{-1}X^T$ is a well defined $n \times n$ matrix.

Next, since it is trivial to verify that the above A satisfies the constraint equation $AX = Y$, hence, it suffices to show that A is a solution of (3.0.1), that is, A has the minimum Frobenius norm among all matrices in the solution space $\mathcal{S} := \{B \in \mathbf{R}^{n\times n}; BX = Y\}$. To

this end, for any $C \in \mathcal{S}$, let $E = C - A$, then $EX = 0$ and

$$\begin{aligned} \| C \|_F^2 &= \| A + E \|_F^2 \\ &= \| A \|_F^2 + \| E \|_F^2 + 2 \sum_{k=1}^{n} \mathbf{a}_k^T \mathbf{e}_k, \end{aligned} \tag{3.1.2}$$

where $\mathbf{a}_k$ and $\mathbf{e}_k$ denote the k^{th} columns of A^T and E^T, respectively, that is,

$$A^T = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n] \quad \text{and} \quad E^T = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_n].$$

Note that $\mathbf{a}_k^T$ and $\mathbf{e}_k^T$ are the k^{th} rows of A and E, respectively.

It follows from the definition of A and the equation

$$X^T E^T = X^T (C^T - A^T) = 0 \quad \text{or} \quad X^T \mathbf{e}_k = 0, \quad k = 1, 2, \cdots, n,$$

that

$$\mathbf{a}_k^T \mathbf{e}_k = \mathbf{y}_k^T (X^T X)^{-1} X^T \mathbf{e}_k = 0, \quad k = 1, 2, \cdots, n, \tag{3.1.3}$$

where $\mathbf{y}_k$ denote the k^{th} column of Y^T.

Combining (3.1.2) and (3.1.3) yields

$$\| C \|_F^2 = \| A \|_F^2 + \| E \|_F^2 \geq \| A \|_F^2. \tag{3.1.4}$$

Hence, A has the minimum Frobenius norm in $\mathcal{S}$. Moreover, since the equality holds in (3.1.4) if and only if $E = 0$ or $C = A$. Thus A is unique minimizer of $\| \cdot \|_F$. The proof is complete. □

Remark 3.1.1. The matrix $X^+ := X^T (X X^T)^{-1}$ is called the Moore-Penrose generalized inverse of the matrix X (see [27]). Hence the matrix A can be rewritten as $A = Y X^+$.

Remark 3.1.2. The AMSM is the only model we found so far which has an explicit solution. It plays an important role in designing numerical methods and algorithms for other models.

3.1.2 AMSM Algorithm

A natural algorithm for computing the solution A given in (3.1.1) is the followings. The first algorithms is efficient for X with full rank. The second algorithm works for X that is not a full rank.

Algorithm 1:

Step 1: Solve the following matrix equation for $W \in \mathbf{R}^{n\times m}$ by Gaussian elimination or QR factorization

$$WX^TX = Y. \tag{3.1.5}$$

Step 2: Set $A = WX^T$.

Algorithm 2:

Step 1: Compute QR factorization of X.

Step 2: Set $X^+ = R^{-1}Q^T$.

Step 3: Set $A = YX^+$.

3.2 The c_p minimum strength model

We solve c_pMSM using the unconstrained minimization formulation

$$\min_{C_0\in\mathbf{R}^{n\times(n-r)}} \|B_1 - C_0V_0^T\|_{c_p}, \tag{3.2.1}$$

where B_1 is a particular solution of $B_1X = Y$ and V_0 is a part of V in $X^T = U\Lambda V^T$. As mentioned before, there are infinitely many solutions of $B_1X = Y$. In this dissertation, our choice of B_1 for c_pMSM and other models is the solution of AMSM since it is easy to compute and efficient.

The c_p-norm ($1 \le p \le \infty$) is one of the largest family of unitary invariant norms (cf. Appendix A3) [44, 74, 76]. For c_pMSM, there holds a following nice result about the solutions of (3.0.2).

Theorem 3.2.1. The unique biological solution of

$$\min_{B\in\mathbf{R}^{n\times n}} \|B\|_{c_p} \quad \text{subject to } BX = Y \tag{3.2.2}$$

is $B_{c_p} = B_{AMSM} := Y(X^TX)^{-1}X^T$, the solution of AMSM.

Proof. We first replace the constrained minimization problem (3.2.2) by its equivalent unconstrained minimization problem. Write $B = B_{AMSM} - B_0$ and $B_0 = C_0V_0^T$, then

$$\min_{B\in\mathbf{R}^{n\times n}} \|B\|_{c_p} = \min_{C_0\in\mathbf{R}^{n\times(n-m)}} \|B_{AMSM} - C_0V_0^T\|_{c_p}.$$

Next, we show that $C_0 = B_{AMSM}V_0$ and the minimum value is zero. Since $\|B\|_{c_p} = \|B^T\|_{c_p}$, V_0 is an unitary matrix and the c_p-norm is an unitary invariant norm,

$$\begin{aligned}
\min_{B\in\mathbf{R}^{n\times n}} \|B^T\|_{c_p} &= \min_{C_0\in\mathbf{R}^{n\times(n-m)}} \|B_{AMSM}^T - V_0C_0^T\|_{c_p} \\
&= \min_{C_0\in\mathbf{R}^{n\times(n-m)}} \|V_0(V_0^TB_{AMSM}^T - C_0^T)\|_{c_p} \\
&= \min_{C_0\in\mathbf{R}^{n\times(n-m)}} \|V_0^TB_{AMSM}^T - C_0^T\|_{c_p}.
\end{aligned}$$

Therefore, $C_0^T = V_0^TB_{AMSM}^T$, that is, $C_0 = B_{AMSM}V_0$.

Finally, to show $B_{c_p} = B_{AMSM}$, it is sufficient to show $B_0 = 0$. But that is obviously true since $B_0 = C_0V_0^T = B_{AMSM}V_0V_0^T = B_{AMSM}\mathbf{0} = \mathbf{0}$. Therefore, $C_0 = B_{AMSM}V_0 = 0$, hence, $B_{c_p} = B_{AMSM}$. □

Remark 3.2.1. Matrix 2-norm (or the spectral norm), $\|A\|_2 = \sqrt{\lambda_{max}(A^TA)} = \sigma_{max}(A)$, is a special case of the c_p-norm.

- For $p = \infty$, the c_p norm is same as the spectral norm.
- For $p = 2$, L^p matrix norm is same as the spectral norm.

Since matrix 2-norm is a unitary invariant norm [8], then the solution of L^pMSM with $p = 2$ is equal to the solution of AMSM.

3.3 Numerical simulation

In this section, we present four sets of numerical experiments to test and validate the AMSM. This will be done first on randomly generated synthetic networks, and then in a subnetwork of *S. cerevisiae* (yeast) cell-cycle using its time-series microarray data obtained from the National Center for Biotechnology Information (NCBI). Our third test is a synthetic five-gene network in *S. cerevisiae* which was proposed by Cantone *et al.* [15] for benchmarking the reverse-engineering techniques and modeling approaches. Lastly, we use our model to recover nine-gene network of *E. coli* using time-series data reported in Bansal *et al.* [4].

Because the number of time points at which gene expression is measured is usually much less than the number of the genes in the network, it is necessary to test the model (and algorithm) with severely under-sampled data sets (i.e., $m << n$). To quantitatively describe the degree of incompleteness of under-sampled data sets, we define

$$DCR := \frac{m}{n} = \frac{\sharp \text{ of time points } (m)}{\sharp \text{ of genes } (n)},$$

and call this number the *data completeness ratio* of the given data set. Clearly, the smaller DCR, the less complete the data set.

To evaluate performance of our model and numerical algorithm, we use different measurements for the synthetic gene regulatory networks and the real gene regulatory networks. For the synthetic networks, we measure the relative error. For the real networks, we only have the qualitative information about the subnetworks of *S. cerevisiae* and *E. coli*, so the focus of our tests is to recover this qualitative information. We first perform post-processing and then measure the performance of our approach using PPV and Se (see Subsections 3.3.1 and 3.3.2).

1. Synthetic gene regulatory network

We first assess accuracy and efficiency of our AMSM and algorithm on some randomly generated synthetic networks. To design the tests, we randomly generate an $n \times n$ network

matrix A and an $n \times m$ data matrix X and set $Y = AX$ with $m = 100$ and 350 and various m values depending on n. The goal of these tests is to recover A using the proposed model and the algorithm. For each test, we measure the relative error between the exact synthetic network matrix and the recovered network matrix.

2. *Saccharomyces cerevisiae* cell cycle

To validate our AMSM and algorithm, we also perform numerical simulations using real experimental data. We first choose the cell cycle of *S. cerevisiae* (yeast) for the purpose because the gene regulatory network of yeast has been well-understood and documented, see [1, 38]. It was also used early as a benchmark to evaluate various models based on different approaches (cf. [56, 57, 86]). In our test, we use a subnetwork of yeast consisting of 14 genes (i.e. $n = 14$) and its microarray profile at 10 time points (i.e. $m = 10$) (cf. [20]). Hence, the DCR of the data set is 0.714.

3. Synthetic five-gene yeast network

The second choice of our numerical simulation using real experimental data is the yeast synthetic network with five genes that is published in [15] to asses reverse engineering and modeling approaches of GRN. Canton *et al.* constructed a synthetic GRN of five yeast genes, then measured gene expression levels of those five genes in two types of data, time-series data and steady state data. For both types, they performed perturbation experiments by shifting cells from glucose to galactose ("switch-on") and from galactose to glucose ("switch-off"). They also evaluated the proposed reverse engineering and modeling approaches using their data. We evaluate our model and algorithm using both switch-on and switch-off time-series data and compare with other approaches which were reported in [15].

4. Nine-gene *E. coli* network

The last real network test of our model and algorithm is the nine-gene *E. coli* network used in [40]. We use microarray profile at 6 time points (i.e. $m = 5$) reported in [4]. Hence, the

DCR of the data set is 0.556. There is no information of perturbation. Thus, Y is only calculated without the perturbation.

3.3.1 Post-processing

Unlike some earlier works [40, 62, 80, 87], we did not assume any priori information about or make any assumptions on the influence matrix A. So in general our identified influence matrices A are *dense* matrices, i.e., majority of the entries of A are nonzero. On the other hand, we also expect that the magnitudes of a majority of the entries in the identified matrix A are very small because lab experiments have shown that a gene often interacts only with a handful of other genes in a (large) gene regulatory network (unfortunately, we do not know a priori which interacts with which). This means that in most cases the real network matrix A should be *sparse* meaning that majority of the entries in A are zeros. So the network matrix A obtained from a mathematical model and its numerical algorithm must be post-processed. In other words, one needs to figure out a threshold value and then use it as a reference to determine which entries of the computed A are set as zero and which are kept unchanged. So far most thresholding strategies reported in the literature are ad hoc [40, 19, 57, 62, 86], which is also the case in all numerical experiments of this section.

To post-process the computed network matrix, we test various threshold values ranging from the smallest to the largest (in absolute value). We note that the signs of the unfiltered entries are kept unchanged in the process. We then compute PPV and Se (see the next section for their definitions) of the post-processed matrix for every threshold value. Finally, we take the post-processed network with the best overall PPV and Se as the inferred network.

3.3.2 Performance evaluation

As we mentioned in Section 1.3.1, nonzero elements in the influence matrix encode the regulatory interactions (activation and inhibition) and magnitude of elements measures the strength of interaction. Currently, regulatory interactions between genes of an organisms can be identified, however, the strength of interaction is more difficult to measure accurately.

Hence, researchers in the reverse engineering of GRN only focus on evaluating regulatory interactions.

In order to evaluate the performance of GRN inferring techniques, we computed the Positive Predicted Value (PPV) and the Sensitivity (Se) that are introduced in [23, 3, 15, 71]. The inferred network can be expressed as one of three different types of graphs.

- Undirected graph: Indicates interactions between genes.
- Directed graph: Indicates interactions between genes and their directions.
- Signed graph: Indicate interactions between genes, their directions, and effects.

The PPV and Se are computed as follows:

$$PPV = \frac{TP}{TP + FP} \quad \text{and} \quad Se = \frac{TP}{TP + FN}, \tag{3.3.1}$$

where TP is the number of true positives (the number of edges in the real network that are correctly inferred), FP is the number of false positives (the number of inferred edges that are not in the real network) and FN stands for the number of false negatives (the number of edges in the real network that are not inferred).

Each type of graphs can be represented by matrices using 0 and ± 1. Table 3.1 shows the meanings of 0 and ± 1 depending on the type of graphs. We note that the connection matrix of undirected graph is symmetric.

The occurrences of TP, FP, and FN need to be measured differently depending on the type of graphs which is described in the Table 3.2. Let R and I be the connection matrices of the real network and the inferred network, respectively.

Table 3.1: Elements in connection matrix

	-1	0	1
Undirected graph	none	no interaction	interaction
Directed graph	none	no interaction	interaction
Signed graph	inhibition	no interaction	activation

Table 3.2: Occurrences of TP, FP, and FN for different types of graphs

	Undirected and Directed graph	Directed signed graph
TP	$R(i,j) = I(i,j) = 1$	$R(i,j) = I(i,j) = 1$ or $R(i,j) = I(i,j) = -1$
FP	$R(i,j) = 0$ and $I(i,j) = 1$	$I(i,j) = \pm 1$ and $R(i,j) \neq I(i,j)$
FN	$R(i,j) = 1$ and $I(j,i) = 0$	$R(i,j) = \pm 1$ and $R(i,j) \neq I(i,j)$

In our approach, the end product is already a matrix. We convert the resulting matrix into a connection matrix for each type of graphs. For the undirected graph, we change nonzero elements to 1, and then make the matrix symmetric by replacing (i,j) element that is zero but (j,i) element 1 by 1. For the directed graph, non-zero elements are replaced by 1. For the connection matrix of the signed graph, negative and positive elements are replaced by -1 and 1, respectively. For example, let

$$A = \begin{pmatrix} 0 & -1.5 & 0 & 0.5 \\ 0 & 0 & 2.8 & -1.1 \\ 0.2 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

be a hypothetical influence matrix given by a numerical simulation. The connection matrix in the case of undirected graph (CU) is converted as follows:

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \implies CU = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

The connection matrix of the directed graph (CD) and the signed graph (CS) are

$$CD = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } CS = \begin{pmatrix} 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

To compute the occurrences of TP, FP, and FN, the inferred network and the real network need to be represented in each case, then TP, FP, and FN are calculated according to the rules in Table 3.2.

For the synthetic gene regulatory network test, we generate a 3000-gene network for the AMSM and test with different numbers of time points m. We observe that the relative errors of $n = 3000$ case are lower than for $n = 100$ and $n = 350$ for the same DCR, that is, the network with more components is recovered with less number of time points. For example, the error of 3000-gene network is dropped below 10% for $DCR = 0.016$, whereas the errors of 100-gene network and 350-gene network are above 10% (Figure 3.1).

Surprisingly, for the 5-gene yeast network, the results of all models that we present in the dissertation are the same.

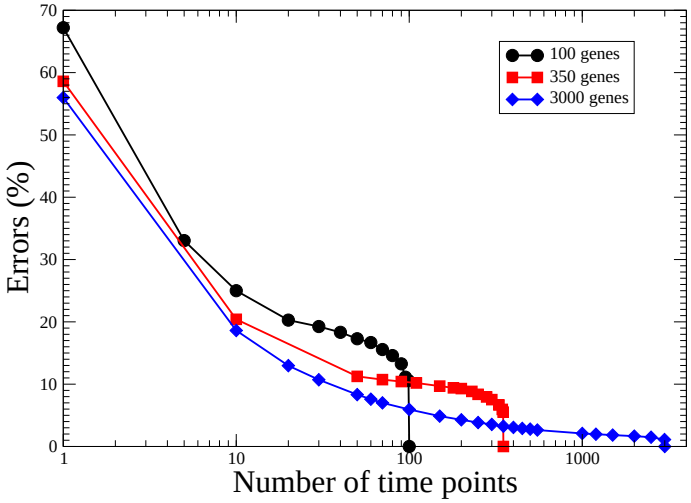


Figure 3.1: AMSM synthetic 100, 350 and 3000 genes networks (x-axis is in log scale)

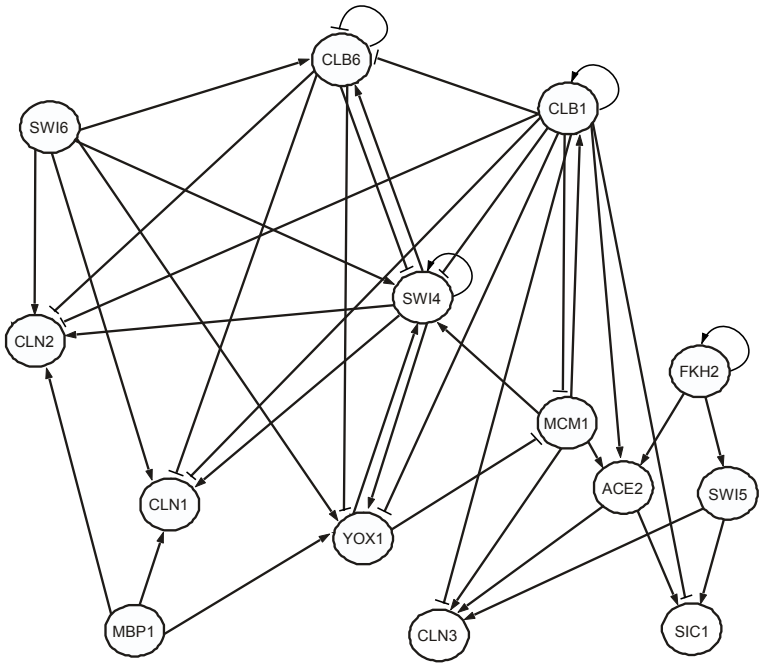


Figure 3.2: True yeast cell cycle network [56]

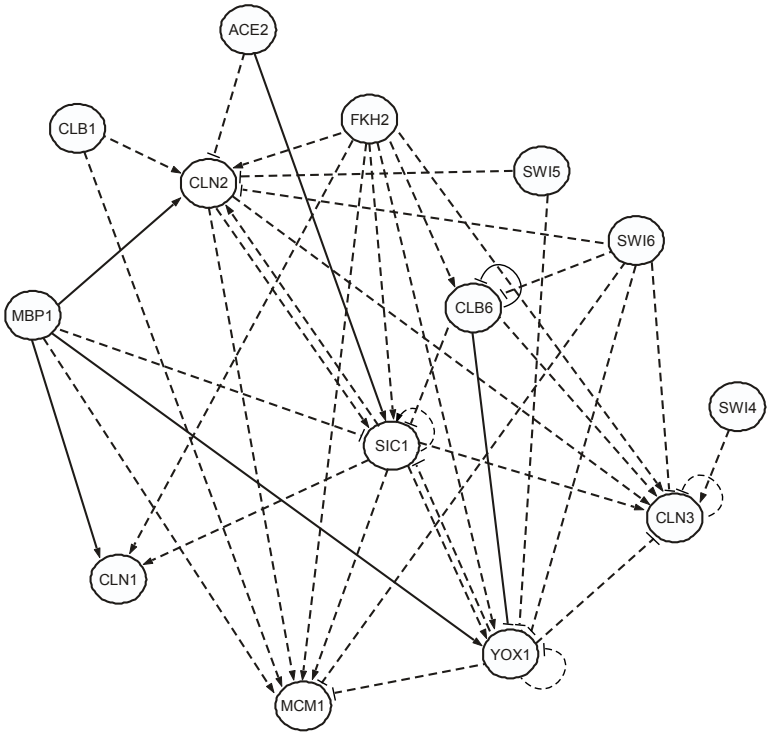


Figure 3.3: Inferred yeast cell cycle network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by AMSM, respectively. PPV^u=0.3, Se^u=0.32, PPV^d=0.29, Se^d=0.29, PPV^s=0.17, Se^s=0.17. ^{u}: undirected graph, ^{d}: directed graph, ^{s}: signed graph.

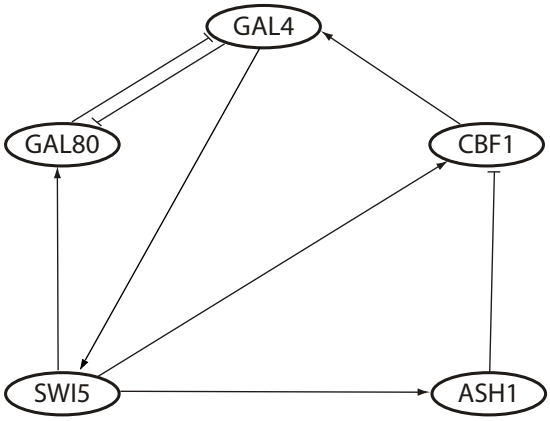


Figure 3.4: Five-gene yeast true network [15]

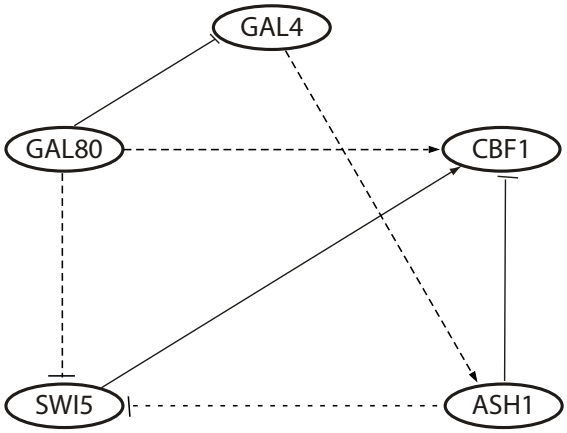


Figure 3.5: Inferred five-gene yeast network using switch on data. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by AMSM using switch on data, respectively. PPV^u=0.71, Se^u=0.57, PPV^d=0.43, Se^d=0.38, PPV^s=0.43, Se^s=0.38.

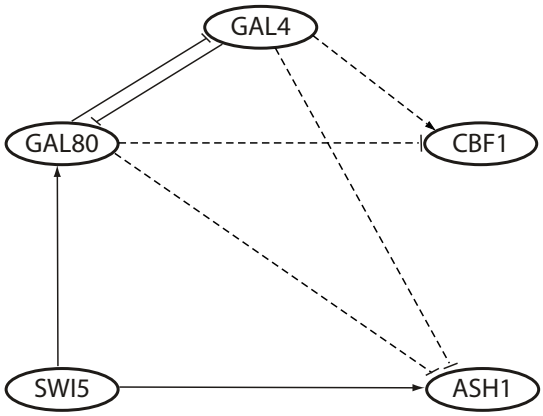


Figure 3.6: Inferred five-gene yeast network using switch off data. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by AMSM, respectively. PPV^u=0.57, Se^u=0.57, PPV^d=0.5, Se^d=0.5, PPV^s=0.5, Se^s=0.5.

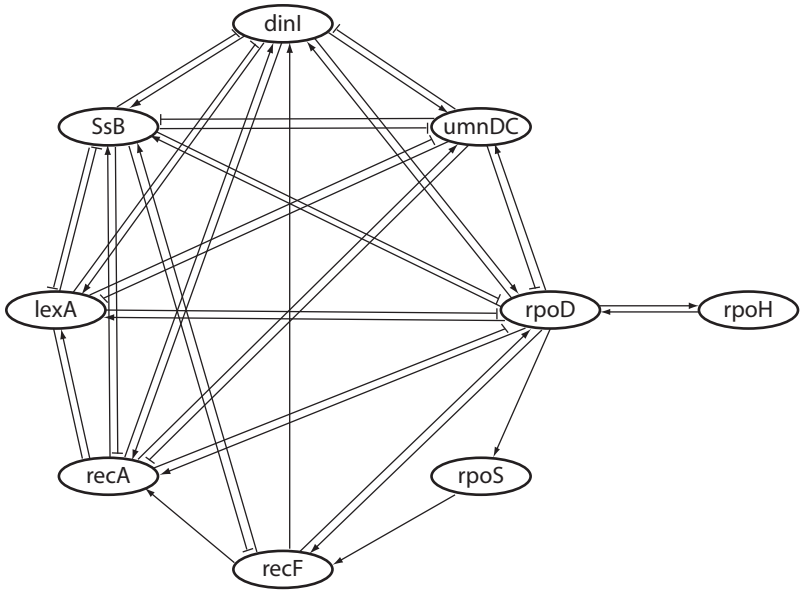


Figure 3.7: Nine-gene true *E. coli* network [4].

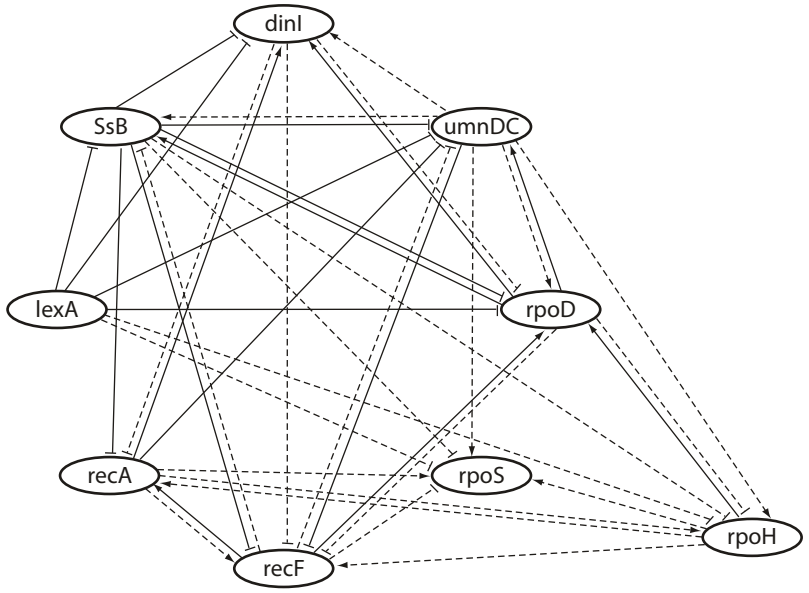


Figure 3.8: Inferred nine-gene *E. coli* network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by AMSM, respectively. PPV^u=0.68, Se^u=0.88, PPV^d=0.67, Se^d=0.65, PPV^s=0.45, Se^s=0.44.

Chapter 4

The Column Minimum Strength and the Row Minimum Strength Models

The Column Minimum Strength Model (CMSM) and the Row Minimum Strength Mode (RMSM) use the functionals $\mathcal{L}(\cdot) = \|\cdot\|_{L^1}$, the L^1-matrix norm and $\mathcal{L}(\cdot) = \|\cdot\|_{L^\infty}$, the L^∞-matrix norm, respectively. For these choices, the abstract variation model (2.1.1) becomes

$$\min_{B\in\mathbf{R}^{n\times n}} \|B\|_{L^1} \quad \text{subject to} \quad BX = Y \tag{4.0.1}$$

and

$$\min_{B\in\mathbf{R}^{n\times n}} \|B\|_{L^\infty} \quad \text{subject to} \quad BX = Y. \tag{4.0.2}$$

Unlike AMSM, there are no closed form solutions for these models. Thus, to construct efficient and fast algorithms becomes necessary to solve these models. To the end, we solve their equivalent unconstrained minimization problems (cf. Chapter 2) with the choice of $B_1 = Y(X^TX)^{-T}X^T$:

$$\min_{C_0\in\mathbf{R}^{n\times(n-r)}} \|B_1 - C_0V_0^T\|_{L^1} \tag{4.0.3}$$

and

$$\min_{C_0 \in \mathbf{R}^{n\times(n-r)}} \|B_1 - C_0 V_0^T\|_{L^\infty}. \tag{4.0.4}$$

The problems now reduce to minimizing a matrix in L^1 and L^∞ norms, such problems have rarely studied in the literature.

In this chapter, we propose two different approaches for solving these problems. The first approach is to compute C_0 column-by-column. For this approach, we use the property,

$$\|B\|_{L^1} = \|B^T\|_{L^\infty}$$

or

$$\|B^T\|_{L^1} = \|B\|_{L^\infty}.$$

We then develop numerical algorithms to approximate C_0 which directly minimizes matrix L^1-norm and L^∞-norm of the matrix $B_1 - C_0 V_0^T$.

The goals of this chapter include establishing the well-posedness of (4.0.3) and (4.0.4) and designing efficient numerical algorithms to compute the solutions of (4.0.3) and (4.0.4) for both approaches. Finally, numerical experiments are presented to show the performance of the CMSM and RMSM and the proposed numerical algorithms for solving the models.

4.1 The column minimum strength model

4.1.1 l^∞-vector norm minimization

To compute C_0 column-by-column approach, we use the property $\|B\|_{L^1} = \|B^T\|_{L^\infty}$. By this property, solving (4.0.3) is equivalent to solve the following problem:

$$\min_{C_0 \in \mathbf{R}^{n\times(n-r)}} \|B_1^T - V_0 C_0^T\|_{L^\infty}. \tag{4.1.1}$$

We propose to compute the solution of (4.1.1), C_0 by minimizing the l_∞-norm of each column of $B^T = B_1^T - V_0 C_0^T$.

Minimizing the l_∞-norm of each column of $B^T = B_1^T - V_0 C_0^T$ approach leads to the following n minimization problems:

$$\min_{\widetilde{\mathbf{c}}_k \in \mathbf{R}^{(n-r)}} \|\widetilde{\mathbf{b}}_k - V_0 \widetilde{\mathbf{c}}_k\|_\infty, \tag{4.1.2}$$

where $\widetilde{\mathbf{b}}_k$ is the k^{th} column of B_1^T and $\widetilde{\mathbf{c}}_k$ is the k^{th} column of C_0^T. Here, for a vector $\mathbf{x} \in \mathbf{R}^n$, $\|\mathbf{x}\|_\infty := \max(|x_1|, |x_2|, \cdots, |x_n|)$. We note that the above minimization is to seek a vector which minimizes the residual of an overdetermined linear system in l_∞-norm. Once $\widetilde{\mathbf{c}}_k$ is computed, we assemble the biological solution by $B = B_1 - C_0 V_0^T$.

The l_∞ minimization of the residual of a linear system has been used in the various areas such as motion computation [75] and tracking a deformable surface [72]. There is no explicit formula to solve (4.1.2). However, it is well-known that l_∞ minimization of the residual of a linear system is equivalent to a linear programming problem [37]. Therefore, any linear programming technique can be used to compute the solution of the l_∞ minimization problem. We adopt this strategy to compute each column of C_0^T.

1. Existence and uniqueness

Since the l_∞-norm is convex, there exists a minimizer of (4.1.2). But the l_∞-norm is not strictly convex. For example, let $\mathbf{x} = [1, 0]^T$ and $\mathbf{y} = [1, 1]^T$ then $\|\mathbf{x}\|_\infty = 1 = \|\mathbf{y}\|_\infty$, $\mathbf{x}+\mathbf{y} = [2, 1]^T$ and $\|\mathbf{x}+\mathbf{y}\|_\infty = 2$, but $\mathbf{x} \neq \mathbf{y}$. Thus, solutions to (4.1.2) may not be unique.

2. Algorithm

CMSM Algorithm 1

We first state the equivalent linear programming problem of (4.1.2) below. It can be shown that has the following equivalent linear programming formulation:

$$\begin{aligned} &\min \ \ t \\ &\text{subject to } \ -t\mathbf{1} \leq \widetilde{\mathbf{b}}_k - V_0 \widetilde{\mathbf{c}}_k \leq t\mathbf{1}, \end{aligned}$$

where $\mathbf{1}$ stands for the vector with constant entry 1. Then, the linear programming problem with variables t, $\widetilde{\mathbf{c}}_k$ is

$$\begin{aligned} &\min \ \overline{\mathbf{d}}^T \overline{\widetilde{\mathbf{c}}_k} \\ &\text{subject to } \overline{V_0}\ \overline{\widetilde{\mathbf{c}}_k} \le \overline{\widetilde{\mathbf{b}}_k} \end{aligned}$$

with

$$\overline{\widetilde{\mathbf{c}}_k} = \begin{pmatrix} \widetilde{\mathbf{c}}_k \\ t \end{pmatrix}, \quad \overline{d} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \overline{A} = \begin{pmatrix} A & -\mathbf{1} \\ -A & -\mathbf{1} \end{pmatrix}, \quad \overline{\widetilde{\mathbf{b}}_k} = \begin{pmatrix} \widetilde{\mathbf{b}}_k \\ -\widetilde{\mathbf{b}}_k \end{pmatrix}.$$

We use the Matlab® built-in function "`linprog`" to solve the above linear programming problem for $k = 1, 2, \cdots, n$.

4.1.2 L^1-matrix norm minimization

The column-by-column approach may destroy relations between columns since it recovers the solution matrix column-by-column. In this section, to overcome this problem, we propose another method which directly minimizes the L^1 matrix norm.

1. Existence and uniqueness

Theorem 4.1.1. There exists a minimizer of (4.0.3).

Proof. The existence of the minimizer of (4.0.3) immediately follows from Theorem 2.1.1. On the other hand, we recall that the matrix L^1-norm is not strictly convex. For example, let

$$X = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ and } Y = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \text{ then } X + Y = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

and $\|X\|_{L^1} = \|Y\|_{L^1} = \frac{1}{2}\|X + Y\|_{L^1}$. However, $X \neq Y$. Therefore, by Definition A.0.2, L^1 norm is not strictly convex. □

In spite of L^1-norm is not strictly convex, we still expect a unique minimizer to (4.0.3)

because of the constraint. Moreover, the strictly convexity is only a sufficient condition for the uniqueness of minimizers. Unfortunately, the uniqueness of the minimizer that also satisfies the constraint has not been proved yet.

2. Algorithm

To construct a numerical algorithm for the L^1-norm minimization, we use the gradient descent method (or the steepest descent method)[11] which is a well-known first-order optimization method. First, let $F(C_0) = \|B_1 - C_0 V_0^T\|_{L^1}$ which is the function of a matrix C_0. At each iteration of the gradient descent algorithm, ∇F must be computed. We approximate ∇F using the finite difference method. For each element of C_0, first compute $C_0(i,j) + h$ and $C_0(i,j) - h$ for any small h and keep the rest of elements unchanged. Let LC_0 and RC_0 denote the two resulting matrices, then

$$\nabla F(C_0)(i,j) \approx \frac{F(LC_0) - F(RC_0)}{2h}.$$

CMSM Algorithm 2

Step 1: Compute the particular solution B_1 using the AMSM Algorithm.

Step 2: Compute C_0 using the gradient descent method.

Choose an initial guess $C_0^{(0)}$, set $F^{(0)} = \|B_1 - C_0^{(0)} V_0^T\|_{L^1}$.

For $l = 0, 1, 2, \cdots, L-1$,

Determine a decreasing step length α.

Set $C_0^{(l+1)} = C_0^{(l)} - \alpha \nabla F(C_0^{(l)})$.

Step 3: Set $B = B_1 - C_0^{(L)} V_0^T$.

4.2 The row minimum strength model

4.2.1 l_1-vector norm minimization

For column-by-column approach, we use the property $\|B\|_{L^\infty} = \|B^T\|_{L^1}$. By this property, (4.0.4) is equivalent to the following problem:

$$\min_{C_0 \in \mathbf{R}^{n\times(n-r)}} \|B_1^T - V_0 C_0^T\|_{L^1}. \tag{4.2.1}$$

Like CMSM column-by-column approach, we propose to compute the solution of (4.2.1) by minimizing the l_1-norm of each column of $B_1^T - V_0 C_0^T$.

During last a few decades, vector l_1-norm has been used as a sparsity-promoting functional in various area such as approximation, compression, and statistical estimation [21, 16, 31, 32, 73, 79]. It has been widely used to approximate a solution to an overdetermined linear system by minimizing the l_1-norm of the residual error since l_1-norm solution is robust to large data errors in the system [7]. Applications and numerical techniques of minimizing vector l_1-norm dramatically increased in the last ten years [33, 30, 14, 13]. Minimizing the residual error of the linear system in l_1-vector norm is especially used to recover the sparsity solution of the system [33]. The sparsity property of solutions is important for GRN problem since the inferred network matrix is meant to be sparse by the "small-world" property of GRN. This motivates us to adopt the l_1-vector norm minimization to (4.2.1).

Minimizing the l_1-norm of each column of $B^T = B_1^T - V_0 C_0^T$ approach leads to the following n minimization problems:

$$\min_{\widetilde{\mathbf{c}}_k \in \mathbf{R}^{(n-r)}} \|\widetilde{\mathbf{b}}_k - V_0 \widetilde{\mathbf{c}}_k\|_1, \tag{4.2.2}$$

where $\widetilde{\mathbf{b}}_k$ is the k^{th} column of B_1^T and $\widetilde{\mathbf{c}}_k$ is the k^{th} column of C_0^T. Here, for a vector $\mathbf{x} \in \mathbf{R}^n$, $\|\mathbf{x}\|_1 := \sum_{i=1}^n |x_i|$. We note that the above minimization is to seek a vector which minimizes the residual of an overdetermined linear system in l_1-norm. Once $\widetilde{\mathbf{c}}_k$ is computed, we assemble the biological solution by $B = B_1 - C_0 V_0^T$.

There are two well-known methods for computing the solution of (4.2.2), they are "Iterative Reweighted Least Squares" (IRLS) and "Linear Programming" methods. It had been shown that the l_1-norm minimization to an overdetermined linear system problem can be expressed as a linear programming problem and solved by linear programming techniques [5, 6, 79]. Linear programming techniques, however, requires the use of a large amount computer memory. Since 1970s, more sufficient IRLS algorithm was developed to solve l_p-norm minimization problem, in particular for $1 \leq p \leq 2$ [25].

1. Iterative reweighted least squares

For given a matrix and vectors $A \in \mathbf{R}^{n\times m}$, $\mathbf{x} \in \mathbf{R}^m$, and $\mathbf{y} \in \mathbf{R}^n$, consider the linear system

$$A\mathbf{x} = \mathbf{y}. \tag{4.2.3}$$

Then the least l^p-norm ($1 \leq p \leq 2$) solution to (4.2.3) is $\mathbf{x} \in \mathbf{R}^m$ such that

$$\mathcal{N}(\mathbf{x}) = \min_{\mathbf{z}\in\mathbf{R}^m} \mathcal{N}_p(\mathbf{z}) \quad for \quad 1 \leq p \leq 2,$$

where

$$\mathcal{N}_p(\mathbf{z}) = \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{m} a_{ij} z_j \right|^p.$$

To derive the normal equations, we set

$$\frac{\partial \mathcal{N}_p}{\partial z_k} = -p \sum_{i=1}^{n} a_{ik} \text{sign}(r(i)) |r(i)|^{p-1} \quad \text{for } k = 1, \cdots, m,$$

where $r(i) = y_i - \sum_{j=1}^{m} a_{ij} z_j$. Then,

$$\frac{\partial \mathcal{N}_p}{\partial z_k} = -p \sum_{i=1}^{n} a_{ik} r(i) |r(i)|^{p-2} = 0.$$

Now, we replace $r(i)$ by its expression to get

$$\sum_{i=1}^{n} a_{ik}|r(i)|^{p-2}\Big(\sum_{j=1}^{m} a_{ij}z_j\Big) = \sum_{i=1}^{n} a_{ik}|r(i)|^{p-2}y_i,$$

or

$$\sum_{j=1}^{m}\Big(\sum_{i=1}^{n} a_{ik}|r(i)|^{p-2}a_{ij}\Big)z_j = \sum_{i=1}^{n} a_{ik}|r(i)|^{p-2}y_i \quad \text{for } k = 1, \cdots, m.$$

The above m normal equations can be expressed in the matrix form as:

$$\mathrm{A}^T\mathrm{W}\mathrm{A}\mathbf{z} = \mathrm{A}^T\mathrm{W}\mathbf{y}, \tag{4.2.4}$$

where

$$\mathrm{W} = \begin{pmatrix} w(1) & & & \\ & w(2) & & 0 \\ 0 & & \ddots & \\ & & & w(n) \end{pmatrix},$$

and $w(i) = |r(i)|^{p-2}$, and $r(i) = b_i - \sum_{j=1}^{m} a_{ij}z_j$ for $i = 1, 2, \cdots, n$. The system (4.2.4) is implicit and non-linear since $r(i)$ and W depend on the unknown vector $\mathbf{z}$. We note that for $p = 2$, W is the identity matrix and (4.2.4) is the normal system of the least squares method.

Numerical algorithm for IRLS approximates $\mathbf{z}$ iteratively. At each iteration, the IRLS algorithm solves the following linear system for $\mathbf{z}^k$,

$$\mathrm{A}^T\mathrm{W}^k\mathrm{A}\mathbf{z}^{k+1} = \mathrm{A}^T\mathrm{W}^k\mathbf{y}, \quad k = 0, 1, 2, \cdots.$$

Since W depends on the unknown vector $\mathbf{z}^k$, we choose the initial condition z^0 such that $W^0 = I$ for the initial condition. The diagonal matrix W^k for $k = 1, 2, \cdots$ is formed with the residuals of k^{th} iteration. The convergence of this algorithm is guaranteed under the following two conditions [12]. First, $w(i)$ is non-increasing in $|r(i)|$. Second, $w(i)$ is bounded

for all i. The first condition is true for $p \leq 2$. To satisfy the second condition, Huber [52] replaced $w(i) = |r(i)|^{p-2}$ by

$$w(i) = \begin{cases} |r(i)|^{p-2} & \text{if } |r(i)| > \epsilon, \\ \epsilon^{p-2} & \text{if } |r(i)| \leq \epsilon \end{cases}$$

for any small $\epsilon > 0$.

2. Existence and uniqueness

Since any vector norm is a convex function, then the l_1- norm is a convex function. Hence, problem (4.2.2) always has a solution by Theorem 2.1.1. Unfortunately, l_1-norm is not strictly convex. It can be shown by the following example. Let $\mathbf{x} = [1, 0]^T$ and $\mathbf{y} = [0, 1]^T$ then $\|\mathbf{x}\|_1 = 1 = \|\mathbf{y}\|_1$, $\mathbf{x} + \mathbf{y} = [1, 1]^T$ and $\|\mathbf{x} + \mathbf{y}\|_1 = 2$, but $\mathbf{x} \neq \mathbf{y}$. Thus, solutions to (4.2.2) may not be unique. In 1988, Zhang *et al.* [89] introduced the limiting solution of l_1 minimization. That is, they remove non-uniqueness of l_1 solution by choosing a solution that is equal to the limit of $\mathbf{x}_p$ as p approaches to 1 from above. Specifically,

$$\mathbf{x}_{1^+} := \lim_{p \to 1^+} \mathbf{x}_p, \tag{4.2.5}$$

where for $A \in \mathbf{R}^{n \times m}$ and $\mathbf{y} \in \mathbf{R}^n$ $(n \geq m)$, x_p is defined by

$$\mathbf{x}_p := \arg\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_p, \qquad p \geq 1. \tag{4.2.6}$$

They showed the existence and uniqueness of the limiting solution which resides in the l_1 solution set.

To find the minimizer of (4.2.2), we apply the IRLS algorithm since it is faster and approximate the unique limiting solution by letting p tend to 1.

3. Algorithm

RMSM Algorithm 1

Step 1: Compute the particular solution B_1 using the AMSM Algorithm .

Step 2: Compute SVD of X^T: $X^T = U\Lambda V^T$.

Step 3: Compute C_0 column-by-column using the IRLS algorithm.

Set $Z_k^{(0)} = I_n$.

For $l = 1, 2, \cdots, L$, do the following:

For $k = 1, 2, \cdots, n$, set

$\mathbf{r}_k^{(l)} = \widetilde{\mathbf{a}}_k^{(l)} - V_0\widetilde{\mathbf{c}}_k^{(l)}$,

$Z_k^l = \mathrm{diag}(\mathbf{r}_k^{(l)})$,

$\widetilde{\mathbf{c}}_k^{(l+1)} = (V_0^t Z_k^{(l)} V_0)^{-1}(V_0^t Z_k^{(l)}\widetilde{\mathbf{a}}_k^{(l)})$.

Step 4: Set $C_0^T = [\widetilde{\mathbf{c}}_1^{(L)}, \widetilde{\mathbf{c}}_2^{(L)}, \cdots, \widetilde{\mathbf{c}}_n^{(L)}]$ and $B = B_1 - C_0 V_0^T$.

4.2.2 L^∞-matrix norm minimization

Although l_1-vector norm minimization is a nice method to approximate the sparse solution, it may destroy relations between columns since it recovers the solution matrix column-by-column. To overcome this problem, we construct an algorithm to directly minimize L^∞ as we proposed for CMSM algorithm 2.

1. Existence and uniqueness

Theorem 4.2.1. There exists a minimizer of (4.0.4).

Proof. The existence of the minimizer of (4.0.4) immediately follows from Theorem 2.1.1. On the other hand, the matrix L^∞-norm is not strictly convex. We can show this using the example of Theorem 4.1.1 by transposing matrices X and Y. □

Although we have not proved the uniqueness of the minimizer that satisfies the constraint, yet, we still expect a unique minimizer with the same reasons for CMSM.

2. Algorithm

Similar to the situation of the CMSM, our numerical algorithm for the RMSM to be introduced below is also of the gradient descent type. To construct a numerical algorithm for L^∞-norm minimization using the gradient descent method, we let $F(C_0) = \|B_1 - C_0 V_0^T\|_{L^\infty}$. Then, we compute $\nabla F(C_0)$ using the finite difference method as it was computed in CMSM algorithm 2.

RMSM Algorithm 2

Step 1: Compute the particular solution B_1 using the AMSM Algorithm.

Step 2: Compute C_0 using the gradient descent method.

Choose an initial guess $C_0^{(0)}$, set $F^{(0)} = \|B_1 - C_0^{(0)} V_0^T\|_{L^\infty}$.

For $l = 0, 1, 2, \cdots, L-1$,

Determine a decreasing step length α.

Set $C_0^{(l+1)} = C_0^{(l)} - \alpha \nabla F(C_0^{(l)})$.

Step 3: Set $B = B_1 - C_0^{(L)} V_0^T$.

4.3 Numerical simulation

We assess CMSM1, CMSM2, RMSM1, and RMSM2 on a randomly generated synthetic networks used for AMSM. We perform CMSM1 and RMSM1 on 100-gene and 350-gene networks and CMSM2 and RMSM2 only on 100-gene network. We also evaluate four models on fourteen-gene yeast cell cycle, five-gene subnetwork and nine-gene *E. coli* subnetwork. For the five-gene yeast subnetwork, the best inferred networks by CMSM1, CMSM2, RMSM1, and RMSM2 are the same as the inferred network by AMSM.

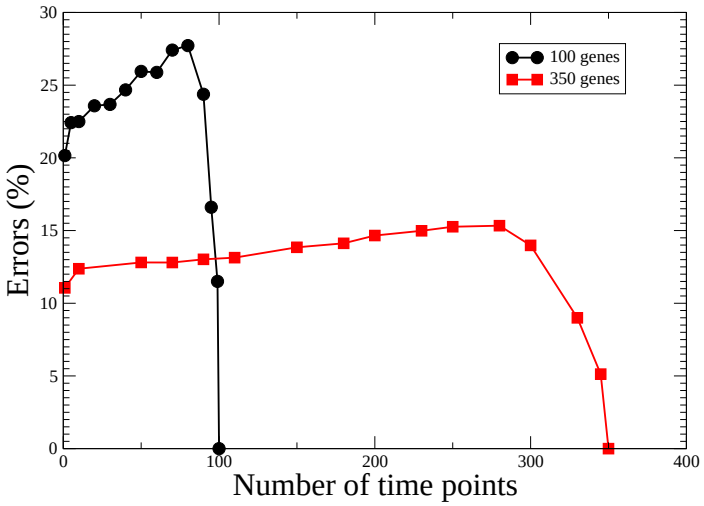


Figure 4.1: CMSM1 on synthetic 100 and 350 genes networks

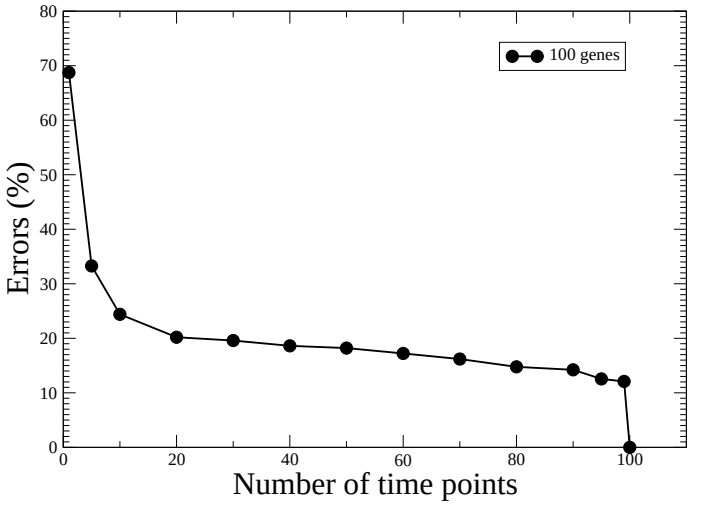


Figure 4.2: CMSM2 on synthetic 100 genes network

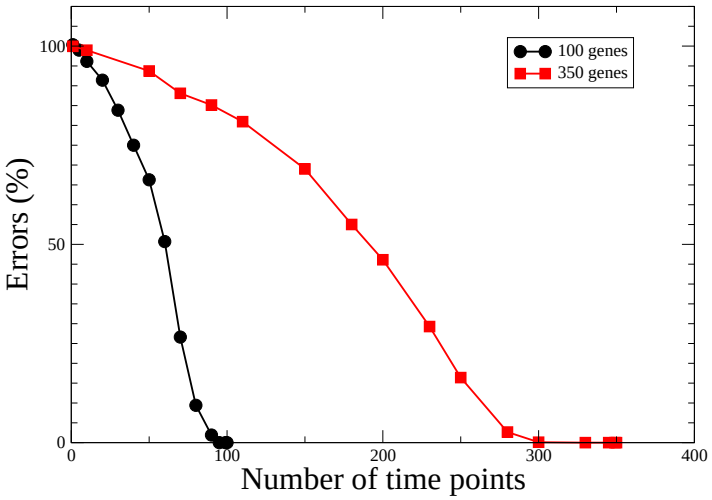


Figure 4.3: RMSM1 on synthetic 100 and 350 genes networks

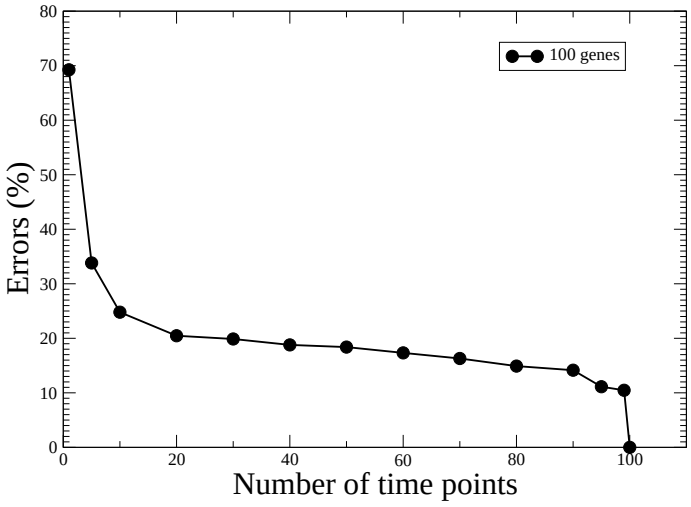


Figure 4.4: RMSM2 on synthetic 100 genes network

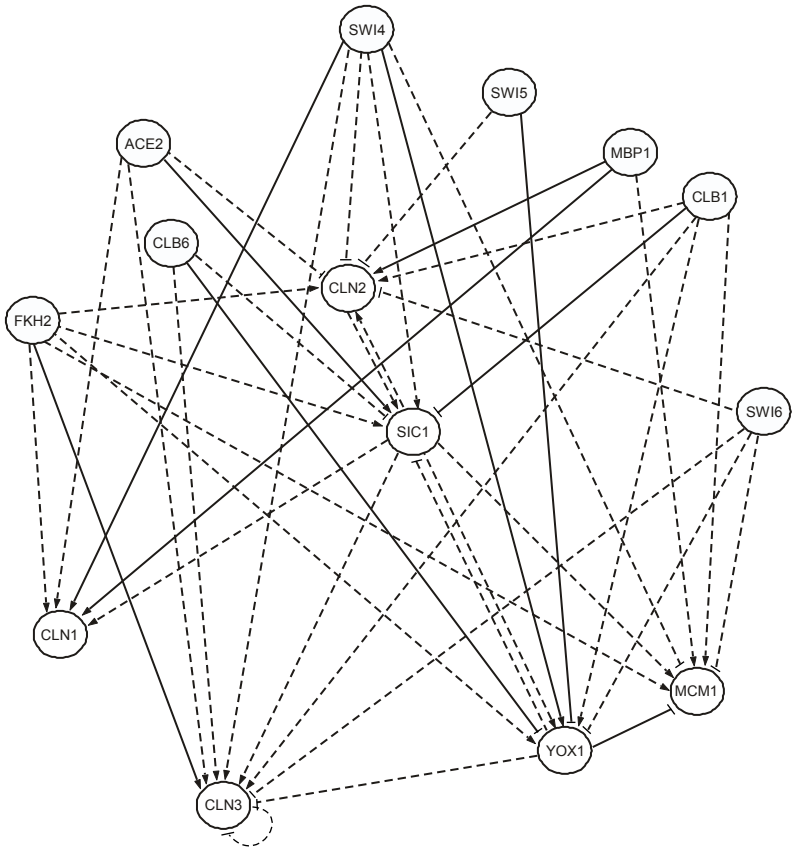


Figure 4.5: Inferred yeast cell cycle network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by CMSM1, respectively. PPV^u=0.42, Se^u=0.47, PPV^d=0.37, Se^d=0.39, PPV^s=0.21, Se^s=0.22.

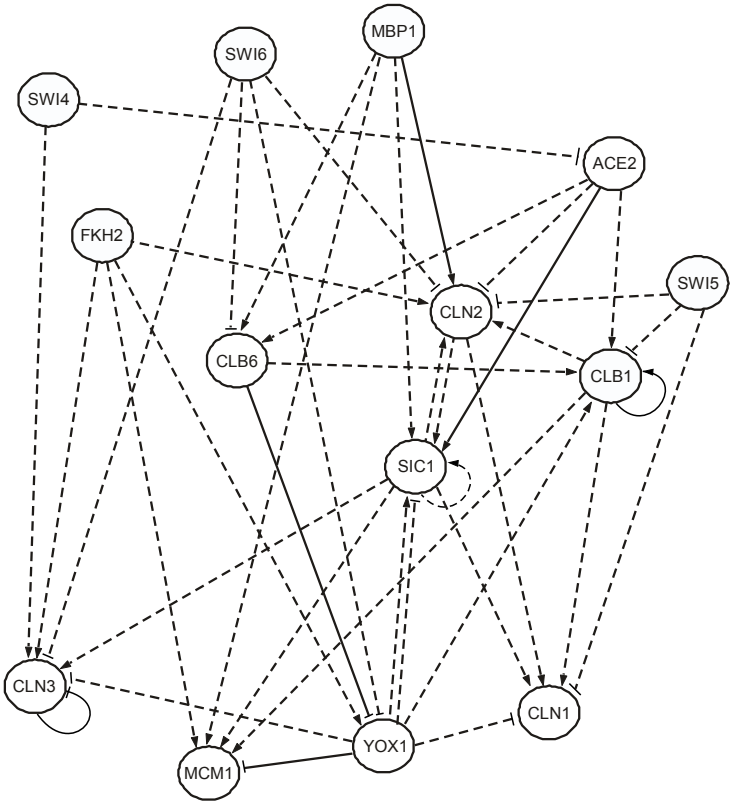


Figure 4.6: Inferred yeast cell cycle network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by CMSM2, respectively. PPV^u=0.36, Se^u=0.36, PPV^d=0.27, Se^d=0.27, PPV^s=0.12, Se^s=0.12.

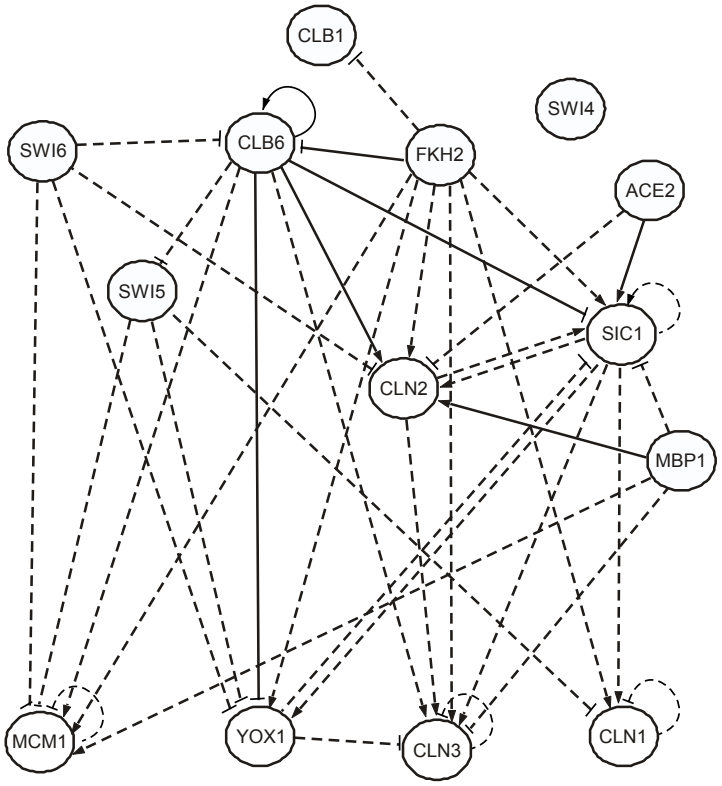


Figure 4.7: Inferred yeast cell cycle network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by RMSM1, respectively. PPV^u=0.21, Se^u=0.12, PPV^d=0.2, Se^d=0.2, PPV^s=0.13, Se^s=0.12.

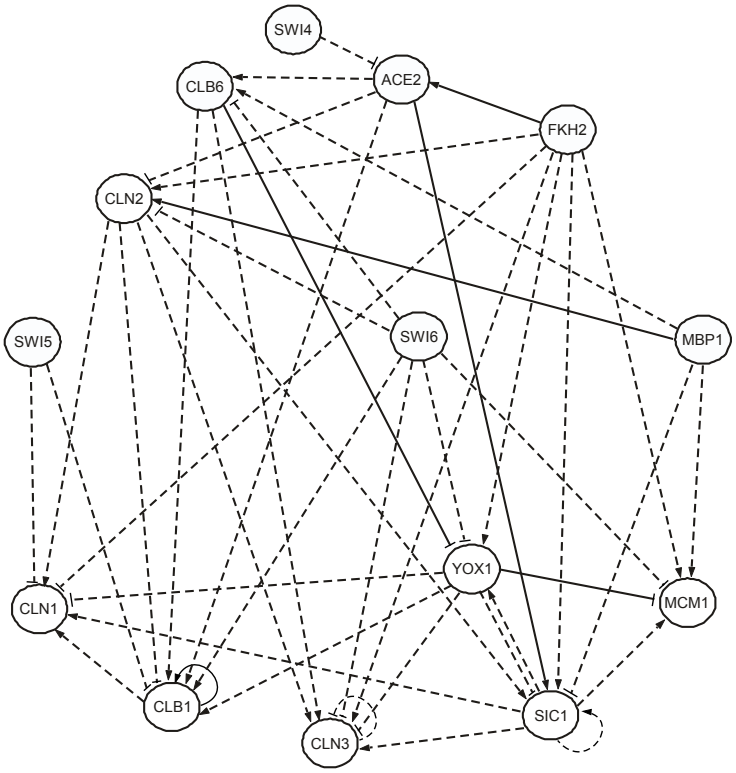


Figure 4.8: Inferred yeast cell cycle network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by RMSM2, respectively. PPV^u=0.33, Se^u=0.38, PPV^d=0.23, Se^d=0.24, PPV^s=0.14, Se^s=0.15.

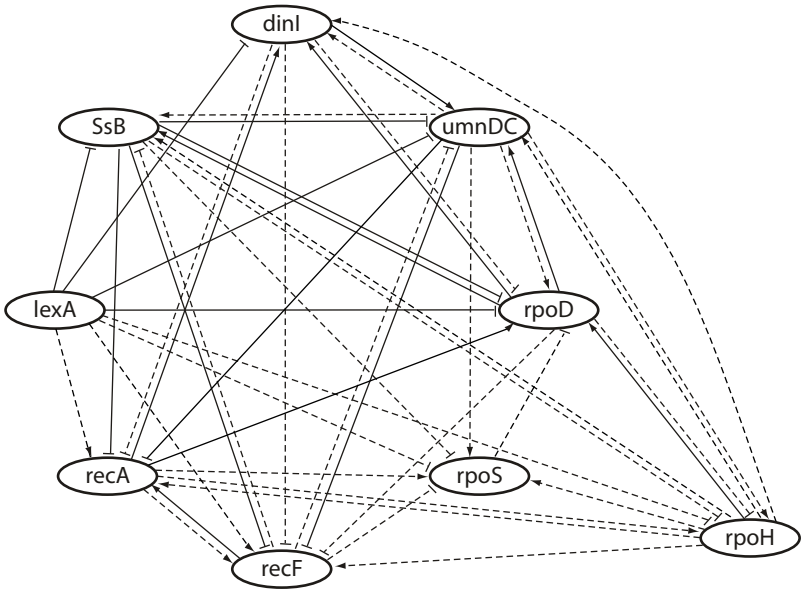


Figure 4.9: Inferred nine-gene *E. coli* network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by CMSM1, respectively. PPV^u=0.68, Se^u=0.96, PPV^d=0.6, Se^d=0.65, PPV^s=0.38, Se^s=0.42.

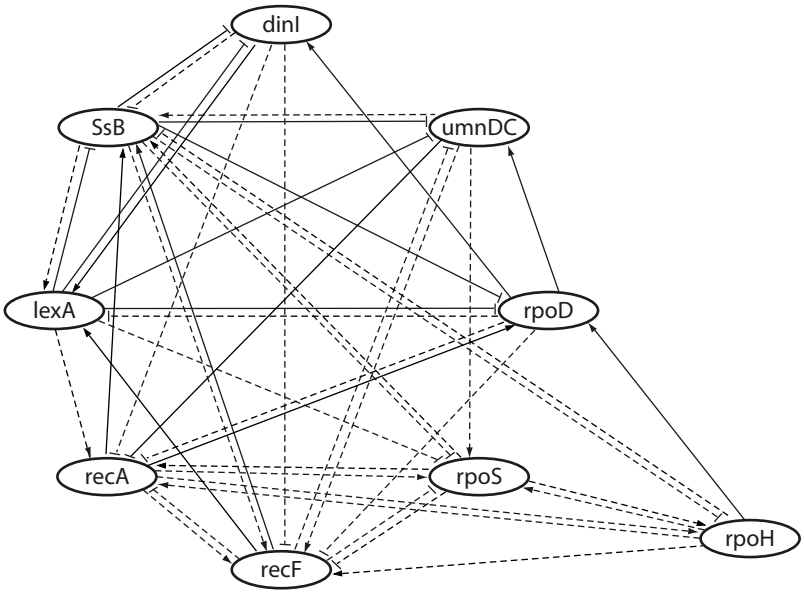


Figure 4.10: Inferred nine-gene *E. coli* network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by CMSM2, respectively. PPV^u=0.73, Se^u=0.92, PPV^d=0.64, Se^d=0.67, PPV^s=0.36, Se^s=0.37.

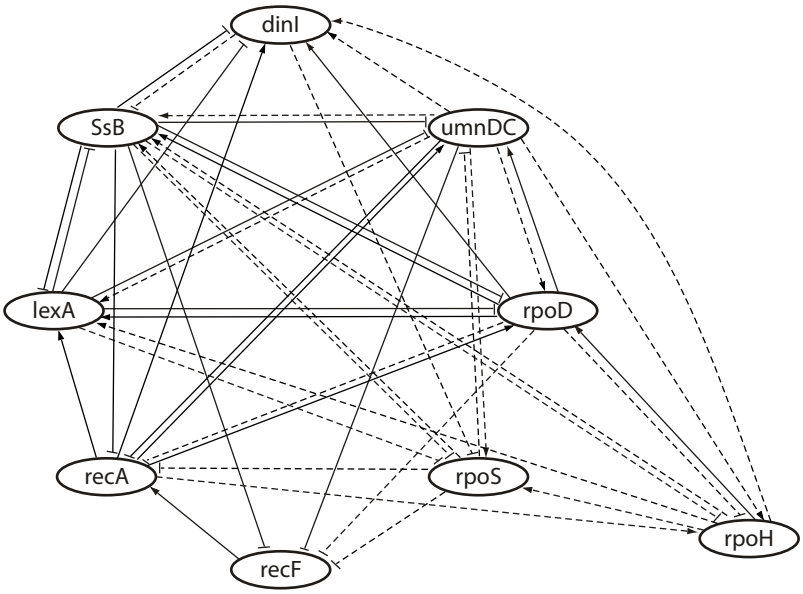


Figure 4.11: Inferred nine-gene *E. coli* network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by RMSM1, respectively. PPV^u=0.66, Se^u=0.88, PPV^d=0.69, Se^d=0.72, PPV^s=0.49, Se^s=0.51.

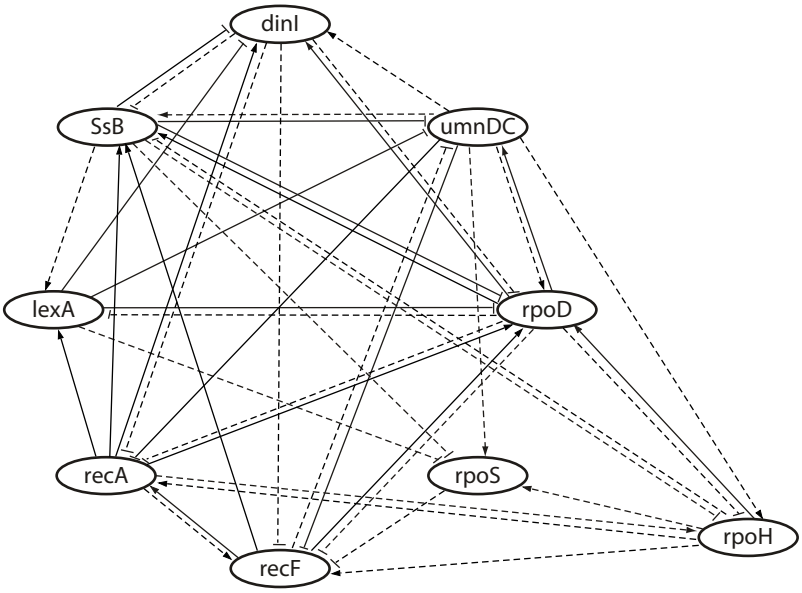


Figure 4.12: Inferred nine-gene *E. coli* network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by RMSM2, respectively. PPV^u=0.73, Se^u=0.92, PPV^d=0.73, Se^d=0.74, PPV^s=0.43, Se^s=0.44.

Chapter 5

The L^p Minimum Strength Model

The L^p Minimum Strength Model (L^pMSM) uses the functional $\mathcal{L}(\cdot) = \|\cdot\|_{L^p}^p$ for $1 < p < \infty$ ($p \neq 2$), where $\|\cdot\|_{L^p}$ is the matrix L^p-norm. For $p = 2$, the matrix L^p-norm is same as the spectral norm which is an unitary invariant norm (cf. Chapter 3). Thus, the solution of the L^2MSM is same as that of the AMSM. Therefore, we do not consider $p = 2$ in this family. For this choice, the abstract variation model (2.1.1) becomes

$$\min_{B \in \mathbf{R}^{n\times n}} \|B\|_{L^p}^p \quad \text{subject to} \quad BX = Y. \tag{5.0.1}$$

The L^pMSM model has no explicit solution. Thus, we solve its equivalent unconstrained minimization problem (cf. Chapter 2) with the choice of $B_1 = Y(X^TX)^{-T}X^T$:

$$\min_{C_0 \in \mathbf{R}^{n\times(n-r)}} \|B_1 - C_0V_0^T\|_{L^p}^p, \quad 1 < p < \infty, \quad p \neq 2. \tag{5.0.2}$$

The problem now reduces to minimizing p^{th} power of the matrix L^p-norm. Unfortunately, the matrix L^p-norm (or Hölder p-nrom) has no explicit representation. Thus, to construct an algorithm to compute a minimizer of (5.0.2), we first need to estimate the matrix L^p-norm. To the end, we adopt an algorithm to compute p^{th} power of the matrix L^p-norm from [50]. In this chapter, we first introduce the algorithm of [50], then compute C_0 by directly minimizing matrix L^p-norm of the matrix $B_1 - C_0V_0^T$.

The goal of this chapter is to address the well-posedness of (5.0.2) and develop an efficient algorithm to compute the solution of (5.0.2) utilizing the algorithm for computing the matrix L^p-norm. Numerical experiments are presented as well to evaluate the performance of the L^pMSM and the proposed numerical algorithms for solving the model.

5.1 Existence and uniqueness of minimizers

Theorem 5.1.1. There exists a solution to problem (5.0.2).

Proof. The existence of a minimizer of (5.0.2) follows immediately from Theorem 2.1.1, since the functional $\mathcal{L}(\cdot) = \|\cdot\|_{L^p}^p$ is convex. □

To show the uniqueness we need to show that $\mathcal{L}(\cdot) = \|\cdot\|_{L^p}^p$ is a strictly convex functional. Unfortunately, we have not proved it yet, although our numerical tests suggest so.

5.2 Algorithm

To construct a numerical algorithm for L^p-norm minimization, we use the gradient descent method as we did for L^1-norm minimization (cf. Chapter 4). The only difference is how the matrix norms are evaluated. As mentioned early, there is no explicit formula to compute the matrix L^p-norm. In this section, we first present an algorithm for approximating the matrix L^p-norm.

5.2.1 The matrix L^p-norm estimation

There is no explicit formula to compute the matrix L^p-norm except $p = 1$ and $p = \infty$. In 1947, Boyd first introduced a method to compute the matrix L^p-norm, which is called the power method [9]. Other methods, the one step estimator and iteratively re-weighted 2-norms, were derived and analyzed later in [50]. All three methods aim at solving the following problem:

$$\max_{\mathbf{x}\in\mathbf{R}^{n\times 1}} \|A\mathbf{x}\|_p \quad \text{subject to } \|\mathbf{x}\|_p = 1.$$

Then, with such a minimizer $\mathbf{x}$, $\|A\|_p = \|A\mathbf{x}\|_p$.

1. The power method

The idea of the power method is to find the critical point of

$$F(\mathbf{x}) := \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p}. \tag{5.2.1}$$

It was proved that if $\mathbf{x}$ is a (nonzero) critical point of $F(\mathbf{x})$ and $\sigma = F(\mathbf{x})$, then

$$\psi_q(A^T\psi_p(A\mathbf{x})) = \sigma^{p(q-1)}\mathbf{x},$$

where $\dfrac{1}{p} + \dfrac{1}{q} = 1$ and $\psi_p(\mathbf{x})$ is the vector with components $|x_i|^{p-1}\text{sign}(x_i)$. Next, define the operators S and W by

$$\begin{aligned} S\mathbf{x} &= \psi_q(A^T\psi_p(A\mathbf{x})) \quad \text{if } \mathbf{x} \neq 0, \\ W\mathbf{x} &= \|S\mathbf{x}\|_p^{-1} S\mathbf{x} \qquad \text{if } S\mathbf{x} \neq 0. \end{aligned}$$

The power method generates the sequence $\{\mathbf{x}^{(k)}\}_{k\geq 1}$ by $\mathbf{x}^{(k+1)} = W\mathbf{x}^{(k)}$ with the expectation that $\mathbf{x}^{(k)}$ converges to a critical point of $F(\mathbf{x})$.

2. The one step estimator

The one step estimator uses the idea of condition number estimation and is not an iterative method [50]. It seeks the vector $\mathbf{x}$ such that $\|\mathbf{x}\|_p = 1$ by computing the component of $\mathbf{x}$ in the order $x_1, x_2, \cdots, x_n$.

Suppose that the first $k-1$ components of $\mathbf{x}$ that satisfies $\|\mathbf{x}(1:k-1)\|_p = 1$ have been determined and let $\gamma_{k-1} = \|A(:, 1:k-1)\mathbf{x}(1:k-1)\|_p$, where $\mathbf{x}(1:k-1)$ and $A(:, 1:k-1)$ denote a column vector consisting of x_1 through x_{k-1} of $\mathbf{x}$ and a matrix consisting of the column 1 through the column $k-1$ of A, respectively. Then the next component x_k is determined and at the same time $\mathbf{x}(1:k-1)$ is revised so that $\mathbf{x}(1:k-1)$ gives the next

partial product a larger norm. The algorithm computes each component of $\mathbf{x}$ by computing λ^* and μ^* such that $\|[\lambda^*\mu^*]\|_p = 1$ and $g(\lambda^*, \mu^*) = \max_{\lambda,\mu} g(\lambda, \mu)$, where $g(\lambda, \mu) = \lambda A(:, 1 : k-1)\mathbf{x}(1 : k-1) + \mu A(:, k)$. Then, set $x_k = \mu^*$ and $\mathbf{x}(1 : k-1) \leftarrow \lambda^*\mathbf{x}(1 : k-1)$.

Higham [50] also proposed to combine the power method and the one step estimator. The power method requires the starting vector $\mathbf{x}^{(0)}$ with $\|\mathbf{x}^{(0)}\|_p = 1$. The combined algorithm first computes the starting vector by the one step estimation and then computes $\mathbf{x}$, which maximize $\|A\mathbf{x}\|_p$, using the power method.

In this dissertation, we use the Higham's algorithm to compute the matrix L^p-norm. To compute the solution matrix using the gradient method, we let $F(C_0) = \|B_1 - C_0V_0^T\|_{L^p}^p$, which is a function of the matrix C_0, and compute $\nabla F(C_0)$ using the finite difference method (cf. Chapter 4).

5.2.2 L^pMSM algorithm

L^pMSM Algorithm

Step 1: Compute the particular solution B_1 using the AMSM Algorithm.

Step 2: Compute C_0 using the gradient descent method as follows:

Choose an initial guess $C_0^{(0)}$, set $F^{(0)} = \|B_1 - C_0^{(0)}V_0^T\|_{L^p}^p$.

For $l = 0, 1, 2, \cdots, L-1$,

Determine a decreasing step length α.

Set $C_0^{(l+1)} = C_0^{(l)} - \alpha\nabla F(C_0^{(l)})$.

Step 3: Set $B = B_1 - C_0^{(L)}V_0^T$.

5.3 Numerical simulation

We assess LpMSM on a randomly generated synthetic network with 50 genes since the computation time is too long to test a larger network. We also evaluate LpMSM on fourteen-gene yeast cell cycle, five-gene subnetwork and nine-gene *E. Coli.* subnetwork. For the five-gene yeast subnetwork, the best inferred network by LpMSM is the same as that inferred

by AMSM.

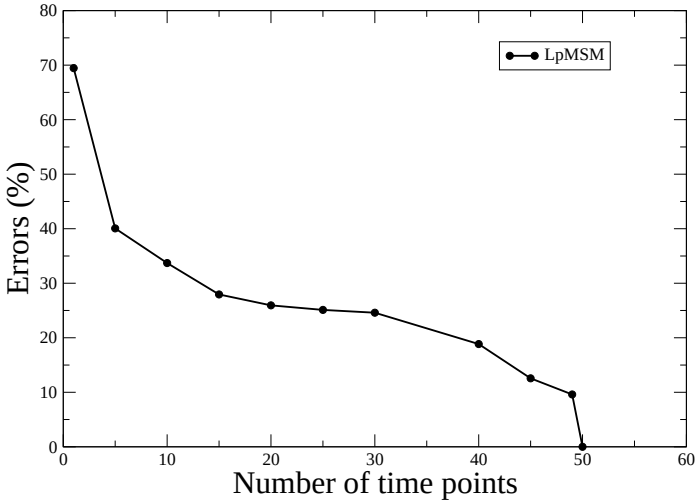


Figure 5.1: LpMSM on synthetic 50 genes network

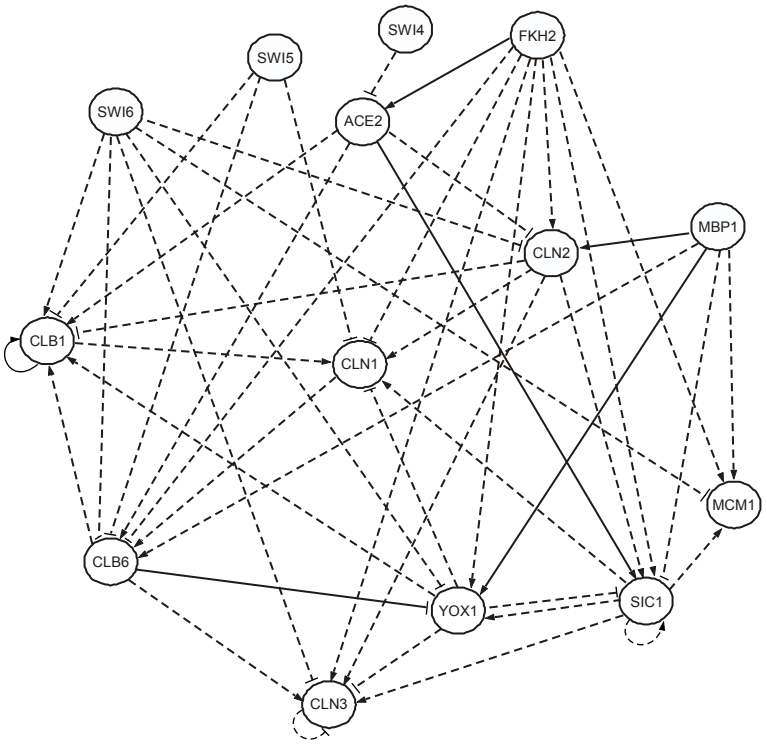


Figure 5.2: Inferred yeast cell cycle network network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by LpMSM, respectively. PPV^u=0.33, Se^u=0.40, PPV^d=0.21, Se^d=0.24, PPV^s=0.13, Se^s=0.15.

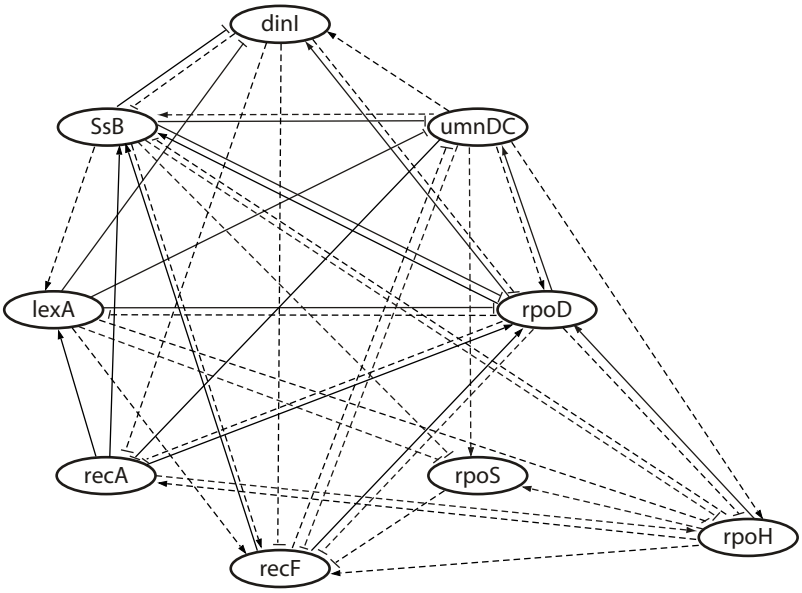


Figure 5.3: Inferred nine-gene *E. coli* network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by LpMSM, respectively. PPV^u=0.71, Se^u=0.92, PPV^d=0.70, Se^d=0.72, PPV^s=0.36, Se^s=0.37.

Chapter 6

The Entrywise Minimum Strength Model

The Entrywise Minimum Strength Model (EMSM) uses the functional $\mathcal{L}(\cdot) = \|\cdot\|_{e_p}^p$ for $1 \leq p \leq \infty$ ($p \neq 2$), where $\|\cdot\|_{e_p}$ is the entrywise matrix p-norm. For $p = 2$, $\|\cdot\|_{e_p}$ is same as the Frobenious matrix norm, which results in the AMSM, see Chapter 3. Thus, we only consider $1 \leq p \leq \infty$ but $p \neq 2$. For this choice, the abstract variation model (2.1.1) becomes

$$\min_{B \in \mathbf{R}^{n\times n}} \|B\|_{e_p}^p \quad \text{subject to} \quad BX = Y. \tag{6.0.1}$$

To construct an efficient and fast algorithm to approximate the minimizer of (6.0.1), we solve its equivalent unconstrained minimization problem with the choice of $B_1 = Y(X^TX)^{-T}X^T$:

$$\min_{C_0 \in \mathbf{R}^{n\times(n-r)}} \|B_1 - C_0V_0^T\|_{e^p}^p, \quad 1 < p < \infty, \quad p \neq 2. \tag{6.0.2}$$

To solve (6.0.2), we again appeal to the gradient descent method and construct an algorithm similar to CMSM Algorithm 2 of Chapter 4. For this model, the function $F(C_0)$ is the entrywise matrix p-norm, that is, $F(C_0) = \|B_1 - C_0V_0^T\|_{e^p}^p$.

In this chapter, we prove the well-posedness of (6.0.2) and describe the details of gradient descent algorithm. We also present numerical experiments to show the performance of the

EMSM and the proposed numerical algorithm.

6.1 Existence and uniqueness of minimizers

Theorem 6.1.1. There exists a solution to problem (6.0.2).

Proof. Once again, the existence follows immediately from Theorem 2.1.1, since $\mathcal{L}(\cdot) = \|\cdot\|_{e^p}^p$ is convex. □

6.2 Algorithm

Similar to the situation of the CMSM and L^pMSM studied in Chapter 4 and 5, our numerical algorithm to be introduced below for the EMSM is also of the gradient descent type. The function to be minimized now is $F(C_0) = \|B_1 - C_0V_0\|_{e^p}^p$. For $1 < p < \infty$, we have an explicit representation for $\nabla F(C_0)$:

$$\nabla F(C_0) = \begin{cases} -p\ (B_1 - C_0V_0^T).^{(p-1)}V_0 & \text{if } p \text{ is even,} \\ -p\ \text{sign}(B_1 - C_0V_0^T).(B_1 - C_0V_0^T).^{(p-1)}V_0 & \text{if } p \text{ is odd,} \end{cases}$$

where $(B_0 - C_0V_0).^{(p-1)}$ denotes the $(p-1)^{th}$ power of each component of the matrix $(B_0 - C_0V_0^T)$. Thus, here, we compute $\nabla F(C_0)$ explicitly and exactly instead of approximately as done in Chapter 4 and 5. For $p = 1$ and $p = \infty$, the functions are not differentiable. Unfortunately, we do not have formulas to compute $\nabla F(C_0)$ for these cases. For numerical simulation, we only consider $1 < p < \infty$.

EMSM Algorithm

Step 1: Compute the particular solution B_1 using the AMSM Algorithm.

Step 2: Compute C_0 using the gradient descent method as follows:

Choose an initial guess $C_0^{(0)}$, set $F^{(0)} = \|B_1 - C_0^{(0)}V_0^T\|_{e^p}^p$.

For $l = 0, 1, 2, \cdots, L-1$,

Determine a decreasing step length α.

Set $C_0^{(l+1)} = C_0^{(l)} - \alpha \nabla F(C_0^{(l)})$.

Step 3: Set $B = B_1 - C_0^{(L)} V_0^T$.

6.3 Numerical simulation

We assess EMSM on some randomly generated synthetic networks which contain 100 and 350 genes. We also evaluate EMSM on fourteen-gene yeast cell cycle, five-gene subnetwork and nine-gene *E. Coli.* subnetwork. For the five-gene yeast subnetwork, the best inferred network by EMSM is the same as that inferred by AMSM.

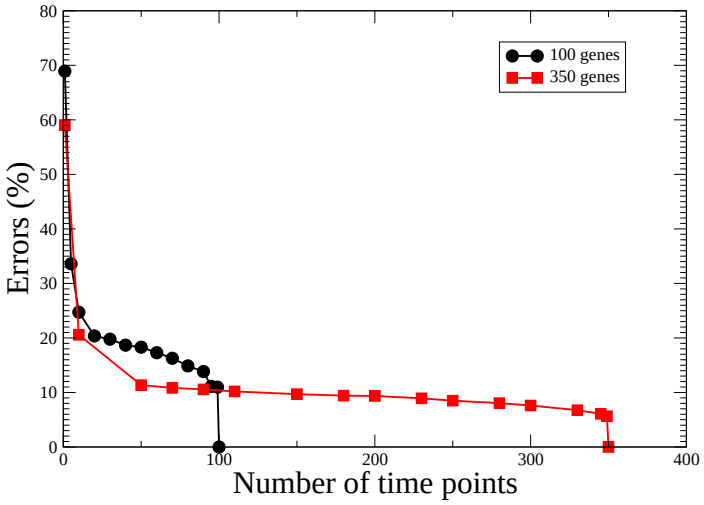


Figure 6.1: EMSM on synthetic 100 and 350 genes networks

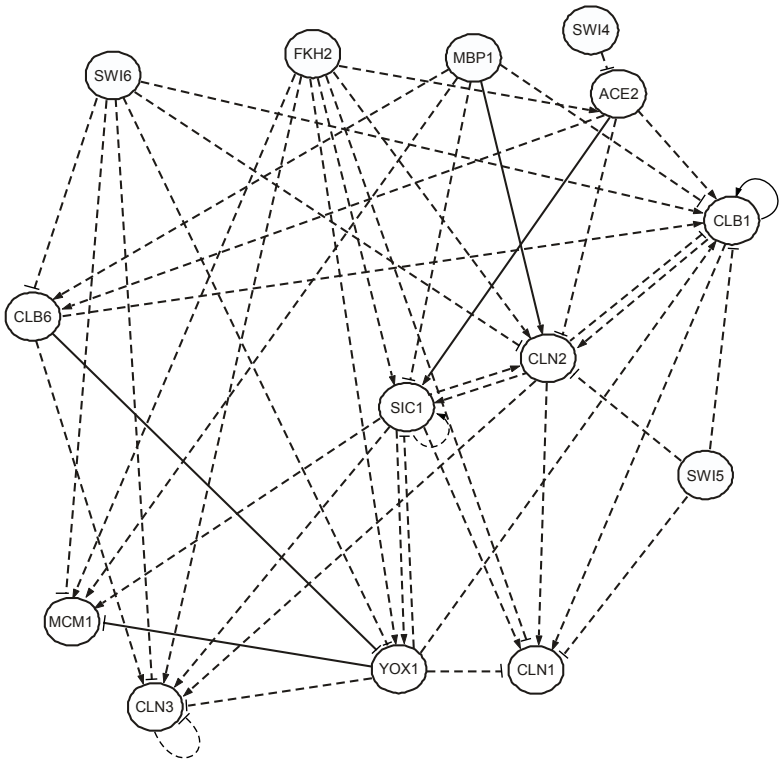


Figure 6.2: Inferred yeast cell cycle network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by EMSM, respectively. PPV^u=0.31, Se^u=0.38, PPV^d=0.23, Se^d=0.27, PPV^s=0.13, Se^s=0.15.

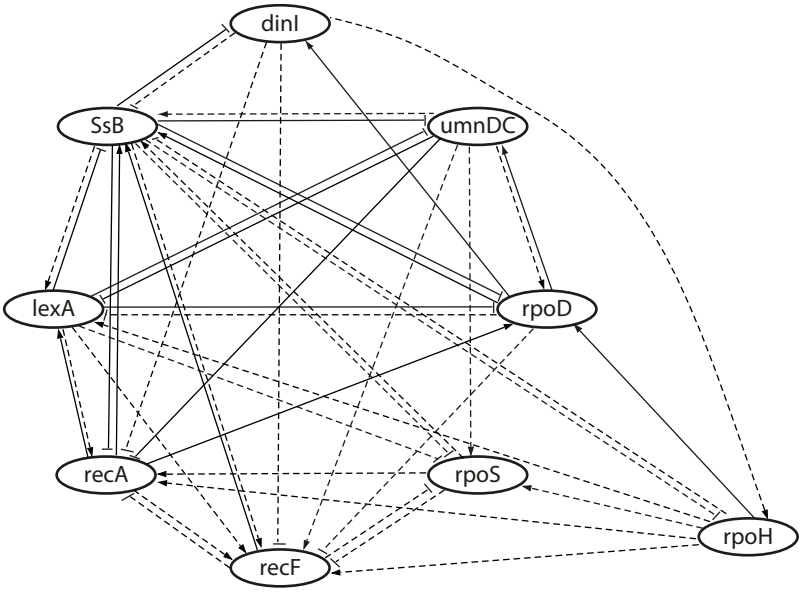


Figure 6.3: Inferred nine-gene *E. coli* network. Solid and dashed lines indicate the correctly inferred and incorrectly inferred edges by EMSM, respectively. PPV^u=0.68, Se^u=0.86, PPV^d=0.74, Se^d=0.65, PPV^s=0.36, Se^s=0.37.

Chapter 7

Stochastic Differential Equation Models and Random Matrix Theory Thresholding Techniques

Uncertainties of microarray data arise in many places such as preparation of samples, measurement errors, and equipment malfunction. Thus, raw microarray data typically contains noise. In order to get better models for gene regulatory networks, those uncertainties should be considered and incorporated into the mathematical models.

We propose stochastic DE models for gene regulatory networks by considering and incorporating those uncertainties into mathematical models. We remark that the idea of using stochastic differential equations to model genetic networks seems new, at least we are not aware of it in the literature, although it is a pretty natural idea.

Similar to the situation of deterministic DE-modeling, the network matrices obtained by stochastic DE models are expected to be dense. As we discussed in Chapter 3, to obtain sparse network matrices, the computed network matrices of stochastic DE models need to be post-processed. Motivated by the works [66, 67] of Luo *et al.*, we propose a novel post-processing method based on the Random Matrix Theory (RMT).

In this chapter, we first present our stochastic DE models and variational framework for gene regulatory network identification. Our stochastic models and variational framework are natural extensions to our deterministic models and framework presented in Chapter 2-6. We then give a brief introduction to RMT, which is followed by a presentation of our RMT-based post-processing method. Finally, numerical experiments are presented for model validation.

7.1 Stochastic differential equation models and stochastic variational framework

To incorporate noise into DE models, we propose to replace the *deterministic differential equation* models (1.3.2) and (1.3.3) by the following *stochastic differential equation* models:

$$\frac{d\mathbf{y}(t,\omega)}{dt} = \mathbf{F}(\mathbf{y}(t,\omega)) + \mathbf{v}(\omega) \quad \omega \in \Omega. \tag{7.1.1}$$

The first order approximation model of (7.1.1) is given by

$$\frac{d\mathbf{x}(t,\omega)}{dt} = \mathrm{A}(\omega)\mathbf{x}(t,\omega) + P(\omega)\mathbf{u}(\omega) \quad \omega \in \Omega, \tag{7.1.2}$$

where ω denotes a sample point and Ω denotes the sample space. More generally, (1.3.2) and (1.3.3) need to be replaced, respectively, by the following Lengvein (stochastic differential) equations

$$\frac{d\mathbf{y}(t,\omega)}{dt} = \mathbf{F}(\mathbf{y}(t,\omega)) + \mathbf{v}(\omega)\xi(t,\omega) \quad \omega \in \Omega, \tag{7.1.3}$$

and

$$\frac{d\mathbf{x}(t,\omega)}{dt} = \mathrm{A}(\omega)\mathbf{x}(t,\omega) + P(\omega)\mathbf{u}(\omega)\xi(t,\omega) \quad \omega \in \Omega, \tag{7.1.4}$$

where $\xi(t,\omega)$ denotes the *white noise* (cf. [59, 70]). We note that in the literature equations (7.1.3) and (7.1.4) are often written formally as

$$d\mathbf{y}(t,\omega) = \mathbf{F}(\mathbf{y}(t,\omega))dt + \mathbf{v}(\omega)dW_t \quad \omega \in \Omega, \tag{7.1.5}$$

and

$$d\mathbf{x}(t,\omega) = \mathrm{A}(\omega)\mathbf{x}(t,\omega)dt + P(\omega)\mathbf{u}(\omega)dW_t \quad \omega \in \Omega, \tag{7.1.6}$$

where $W_t, t \geq t_0$ stands for the Wiener (stochastic) process (cf. [59, 70]).

Like in the case of the deterministic DE models, we assume $P(\omega) = I$ in (7.1.2). Then (7.1.2) becomes

$$\frac{d\mathbf{x}(t,\omega)}{dt} = \mathrm{A}(\omega)\mathbf{x}(t,\omega) + \mathbf{u}(\omega) \quad \omega \in \Omega. \tag{7.1.7}$$

We now use (7.1.7) to formulate our stochastic variational framework for identifying the random network matrix $A(\omega)$. The formulation for (7.1.4) with $P(\omega) = I$ is similar.

We discretize (7.1.7) as it was done for the deterministic DE models. The gene expression data $\mathbf{x}(t,\omega)$ is measured at $m+1$ time points $\{t_j\}_{j=1}^{m+1}$ in a lab experiment. Therefore, we approximate $\dfrac{d\mathbf{x}(t,\omega)}{dt}$ at each time point t_j $(1 \leq j \leq m)$ by

$$\frac{d\mathbf{x}(t_j,\omega)}{dt} \approx \frac{\mathbf{x}(t_{j+1},\omega) - \mathbf{x}(t_j,\omega)}{t_{j+1} - t_j}.$$

After discretization, equation (7.1.7) reduce to

$$A(\omega)X(\omega) = Z(\omega), \quad \omega \in \Omega, \tag{7.1.8}$$

which is the stochastic counterpart part of the deterministic relation (1.3.8), where

$$X(\omega) = [\mathbf{x}(t_1,\omega), \mathbf{x}(t_2,\omega), \cdots, \mathbf{x}(t_m,\omega)],$$
$$Z(\omega) = [\mathbf{z}_1(\omega), \mathbf{z}_2(\omega), \cdots, \mathbf{z}_m(\omega)],$$

and $\{\mathbf{z}_j(\omega)\}_{j=1}^m$ stands for discrete approximations of the rate $(\frac{d\mathbf{x}(t,\omega)}{dt} - \mathbf{u}(\omega))$ at the same m time incidences. As in the deterministic case, equation (7.1.8) has infinitely many solutions $A(\omega)$, and the least squares method for the equation also has infinitely many solutions $A(\omega)$. Hence, a solution selection criterion is needed to pick up the "biological solution".

To extend the deterministic variational framework to the above problem, for each $\omega \in \Omega$, let $\mathcal{S}(\omega) := \{B(\omega) \in \mathbf{R}^{n\times n}; B(\omega)X(\omega) = Z(\omega)\}$. Let $\mathcal{L}$ be a pre-determined (energy) functional on $\mathbf{R}^{n\times n}$, the stochastic analogue of the variational problem (2.1.1) is

$$\min_{B(\omega)\in\mathcal{S}(\omega)} \mathcal{L}(B(\omega)) \quad \omega \in \Omega. \tag{7.1.9}$$

However, problem (7.1.9) is not feasible numerically because the sample space may contain infinitely many samples. One remedy for this is to consider the so-called worst case approach, which looks for the worst outcome that might occur and makes no distinction between outcomes according to the probability distribution function. In this case, the minimization problem is defined as

$$\max_{\omega\in\Omega} \min_{B(\omega)\in\mathcal{S}} \mathcal{L}(B(\omega)). \tag{7.1.10}$$

Another alternative approach is to minimize the expectation value of the random variable $\mathcal{L}(B(\omega))$, the overall energy of $B(\omega)$. The minimization problem in this case is defined as

$$\min_{B(\omega)\in\mathcal{S}(\omega)} E(\mathcal{L}(B(\omega))). \tag{7.1.11}$$

We note that problems (7.1.9) - (7.1.11) are known as *stochastic optimization* problems [58].

In this dissertation, to compute the solution of (7.1.9), we first choose a set of sample points. Then, for each sample point $\omega \in \Omega$, we have a deterministic variational problem as (2.1.1). Next, we solve each deterministic variational problem with a choice of the functional $\mathcal{L}$. Finally, we set the mean of computed minimizers $\{B(\omega_1), B(\omega_2), \cdots, B(\omega_s)\}$ to be the solution of (7.1.9) [24]. The resulted matrix is not a random matrix any more. However, we

expect that all minimizers are close to their mean. Therefore, the mean of all minimizers gives the structure of the inferred matrix, that is, the structure of the network.

7.2 Post-processing of stochastic differential equation models

The resulting solutions from the stochastic differential equation models are almost certainly dense matrices. Thus, they must be post-processed to have the sparse solutions. Luo *et al.* [66, 67] proposed to use random matrix theory for determining the threshold values of correlation matrix of microarray data to discover gene functional modules. We proposed to adopt the post-processing idea of Luo *et al* to determine the threshold values to post-process the solutions of our stochastic DE models.

7.2.1 Random matrix theory (RMT)

In 1928, Wishart first introduced Random Matrix Theory in mathematical statistics studying correlations between different features of a population [85]. Wigner, the theoretical physicists, proposed the concept of statistical distribution of nuclear energy levels in the 1950s [82]. The locations of peaks in nuclear reactions of an atom are called energy levels (see Figure 7.1). The ground state and low lying excited states have been impressively explained. However, at higher excitations, the nuclear states are so dense and the intermixing is so strong. Hence, it is unable to explain the individual states. Nuclear physicists focused on the average properties of nuclear states at higher excitations instead explaining the characteristics of every individual state.

The energy levels of a dynamical system are supposed to be described by the eigenvalues of a Hermitian operator, H, called Hamiltonian [68]. However, H is unknown. Even it is known, it is too complicated to compute eigenvalues. Wigner proposed that the local statistical behavior of energy is identical to the eigenvalues of a large random matrix, a matrix whose elements are random variables with a given probability law. Motivated by Wigner's proposal, random matrix theory has been established to analyze the statistical

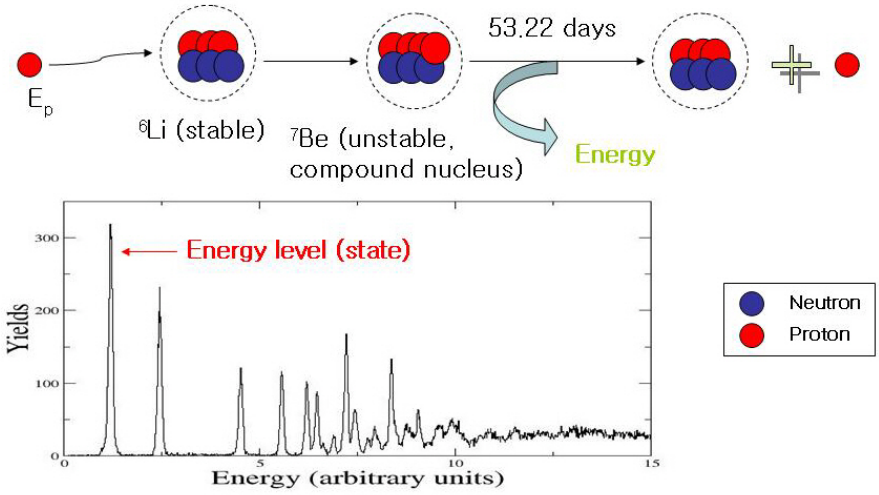


Figure 7.1: Nuclear reaction of Lithium

behavior of the eigenvalues of random matrices such as the eigenvalue density and eigenvalue spacing distribution of random matrices. In 1956, Wigner derived the famous Wigner's surmise [83],

$$P(s) \approx As \exp(-Bs^2)$$

that represented the distribution of the spacing between consecutive eigenvalues of real symmetric random matrices. In 1962, Dyson introduced the following classification of random matrix ensembles which model the Hamiltonians of random dynamical systems [35, 36]:

- Gaussian Orthogonal Ensembles (GOE): the family of real symmetric matrices whose entries are independent and identically distributed (i.i.d.) Gaussian.
- Gaussian Unitary Ensembles (GUE): the family of Hermitian matrices whose entries are i.i.d. Gaussian.
- Gaussian Sympletic Ensembles (GSE): The family of self-dual matrices whose entries are i.i.d. Gaussian.

Later, it has been proved that Wigner's surmise only approximates the nearest neighbor

spacing distribution (NNSD) of eigenvalues of GOE and it is called Wigner-Dyson distribution

$$P_{GOE}(s) \approx \frac{1}{2}\pi s \exp(-\pi s^2/4).$$

Metha and Gaudin derived the exact expression for NNSD of Gaussian ensembles [69, 42]. Their expressions for the distributions are close to Wigner's surmise. Empirical evidence suggests that Wigner's surmise is almost universally applicable.

Since the 1980s, scientists have been studying other mathematical properties of random matrices such as condition numbers and singular values of random matrices. RMT has been successfully applied to various scientific fields such as nuclear physics, quantum physics [68, 47], Riemann Hypothesis, stock market [60], complex networks [2], and biological networks [66, 67, 65].

7.2.2 RMT thresholding techniques

In [66, 67], the authors first construct the correlation matrix of genes using microarray data. The correlation matrix is dense which means that all genes are connected each other. They remove elements (or edges in the network) of the correlation matrix to discover biological networks of genes by using the Girvan and Newman algorithm [43] that gradually deletes the edges. The question is when the deletion has to be stopped.

According to Luo *et al.* [66, 67], the NNSD of eigenvalues of the correlation matrix before deleting elements follows the Poisson distribution,

$$P_{Poisson}(s) = \exp(-s).$$

At each deletion step, they computed the NNSD of eigenvalues of the correlation matrix. They provided the evidence that the NNSD of eigenvalues of the correlation matrix transited from the Poisson distribution to the Wigner-Dyson distribution. They proposed to stop the the deletion procedure when NNSD of eigenvalues reaches to the Wigner-Dyson distribution.

For any post-processing method, the key issue is to determine the right threshold values,

which means when to stop deleting elements. Our hypothesis on post-processing step is that the NNSD of singular values of the "true" gene regulatory network matrix follows the Wigner-Dyson distribution. We choose the cut-off values same as deterministic DE models and compute the NNSD of singular values for each chosen value. When the NNSD of singular values confirms the Wigner-Dyson distribution, we stop the searching process and select the final cut-off value as the threshold value.

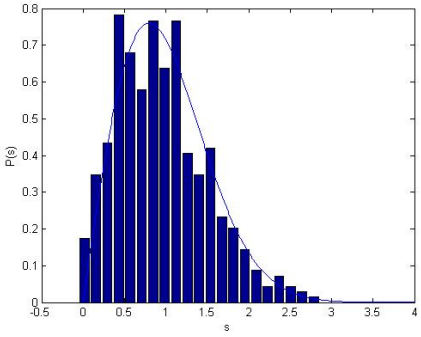
7.3 Numerical simulations

In this section, we present one set of numerical experiments to test and validate the stochastic DE model and RMT based post-processing technique. We choose the functional $\mathcal{L}(\cdot) = \|\cdot\|_{L^\infty}$ with RMSM1. We only test on a randomly generated synthetic network since the RMT is only applicable to large size of matrices and we do not have access to large time-series microarray data sets. The real networks that we have data only consist of less than 15 components. Thus, it is not large enough to apply the RMT.

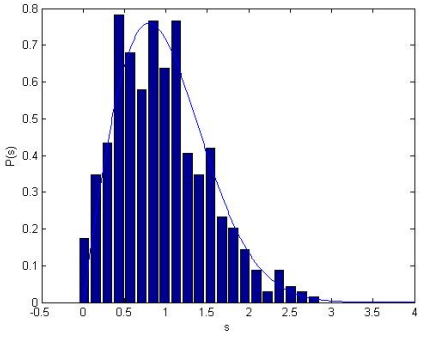
We generate a network matrix A, data matrix X, Y in the same way as we did for the deterministic DE models. To test the stochastic DE model, we add 100 different Gaussian noise to X so that we have 100 noisy data matrices. We, then recover the network matrix A with $m = 100$ and $m = 200$. Lastly, we plot the NNSDs of singular values of the inferred network matrix before post-processing and after post-processing with various theresholding values.

Figure 7.2 and 7.3 show the NNSDs of singular values of inferred network matrices with various thresholding values for $m = 100$ and $m = 200$, respectively. Solid curve represents the Wigner-Dyson distribution and blue bars are the NNSD of singular values. In Figure 7.2, before post-processing, Stage 1 and 2 NNSDs of singular values are close to each others and are a bit off of Winer-Dyson distribution. After Stage 3, NNSDs of singular values are getting closer to the Wigner-Dyson distribution and NNSD of singular values at Stage 5 is the best fit to the Wigner-Dyson distribution, overall.

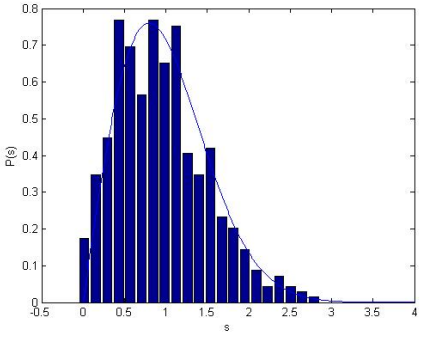
We also observe that the NNSDs of singular values are changed at each stage in Figure 7.3. Stage 1 is the closest to the Winer-Dyson distribution.



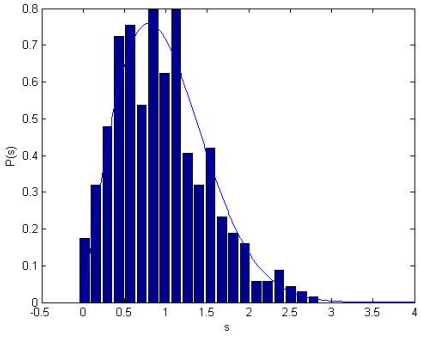
(a) Before post processing



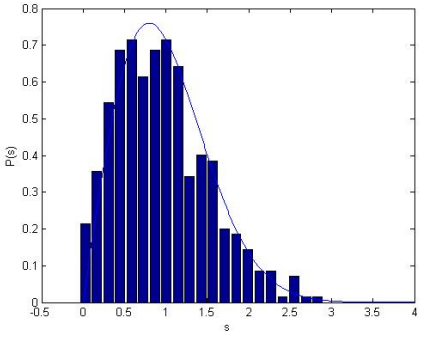
(b) Stage 1



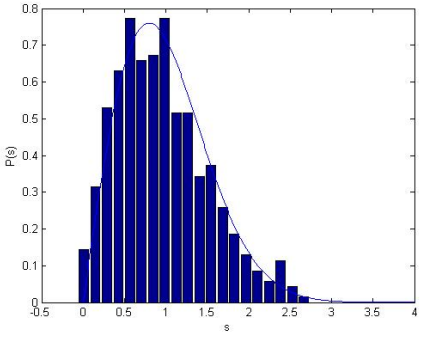
(c) Stage 2



(d) Stage 3

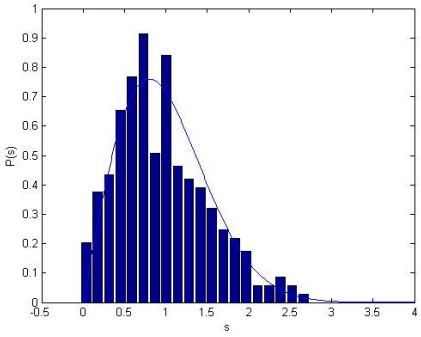


(e) Stage 4

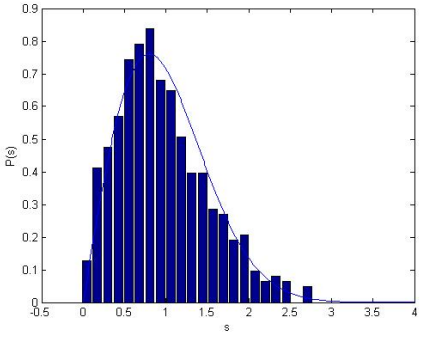


(f) Stage 5

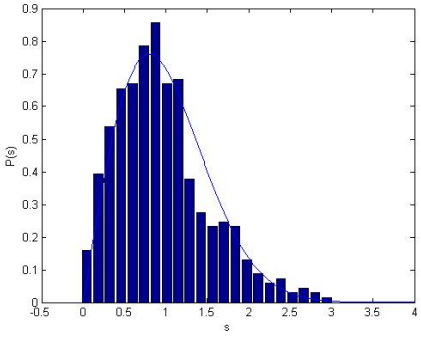
Figure 7.2: NNSDs of singular values of inferred network matrices with $m = 100$



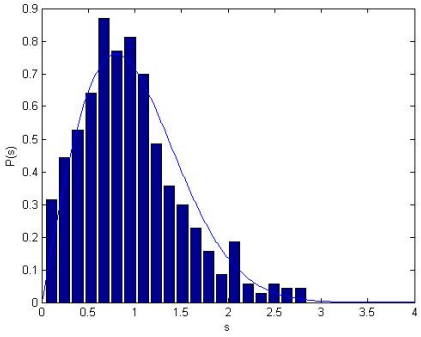
(a) Before post processing



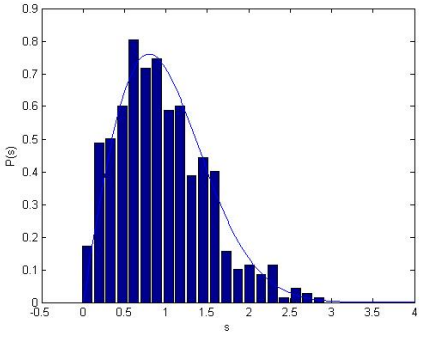
(b) Stage 1



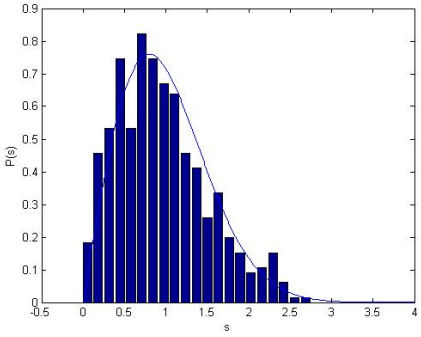
(c) Stage 2



(d) Stage 3



(e) Stage 4



(f) Stage 5

Figure 7.3: NNSDs of singular values of inferred network matrices with $m = 200$

Chapter 8

Conclusion and Future Directions

8.1 Conclusion

We have presented a general variational framework and several specific models within this framework for gene regulatory network identification based on differential equation approach. Closed form solutions or efficient numerical algorithms are also developed for computing the solutions of the variational models. In the framework, no priori structure condition is assumed or imposed on the gene regulatory network to be identified, the network is determined solely by the microarray profiles of the network components, and is identified by a variational principle which minimizes a "biological energy functional". Such a variational principle not only serves as a selection criterion to pick up the right biological solution but also can be regarded as a mathematical description of the "small-world" property of gene regulatory networks which has been observed in lab experiments.

The proposed framework, models, and numerical algorithms are evaluated and tested on both randomly generated synthetic networks and on the benchmark subnetworks of *S. cerevisiae* and *E. Coli*.

8.1.1 Synthetic gene regulatory networks

Figure 8.1 and 8.2 give head-to-head comparisons of the relative errors of all models except LpMSM for $n = 100$ and $n = 350$. For $n = 350$, we only could perform the numerical simulations on AMSM, CMSM1, RMSM1, and EMSM since the computation times of CMSM2, RMSM2, and LpMSM are too long on a large network. Especially, LpMSM is not efficient for a network lager than $n = 100$. Thus, we test LpMSM with $n = 50$ whose result is presented in Chapter 5. Both results show that AMSM, CMSM2, RMSM2, and EMSM solutions are close to each other as expected since our recovered network is an approximation of the same true network. We note that AMSM, CMSM2, RMSM2, EMSM recover the network matrix by minimizing matrix norms, while CMSM1 and RMSM1 recover network matrix column-by-column by minimizing vector norms. Thus, CMSM1 and RMSM1 give slightly different results than other methods. The results also suggest that the accuracy of the models depend on the DCR of the data set. Although similar general patterns of the errors are observed for all models, the AMSM, CMSM2, RMSM2, and EMSM perform better for low DCR data sets. The errors drop significantly when DCR of the data set is just above 0.05 (or 5%) for $n = 100$ and 0.029 (or 2.9%) for $n = 350$, whereas RMSM1 still produces large errors. However, when $DCR \geq 0.8$ (or 80%) for each n, the situation is reversed, that is, the error of the RMSM1 is smaller than those of other models which are pretty much flat. Hence, RMSM1 outperforms the AMSM, CMSM2, RMSM2, and EMSM for high DCR data sets. One "extreme" example is when $n = 350$ and $m = 330$ (hence, $DCR = 0.94$), the RMSM1 essentially recovers the "exact network", but the errors of the AMSM, CMSM1, EMSM change little for $50 \leq m \leq 330$. Overall, the error of CMSM1 changes little for different m. CMSM1 performs best among all models when DCR of the data set is just less than 0.1 (or 10%) for $n = 100$ and 0.029 (or 5%) for $n = 350$. The above test results suggest that the possibility to design a more accurate hybrid model which combines the AMSM, CMSM1, and RMSM1 (using DCR as a switch) and takes the advantage of three models.

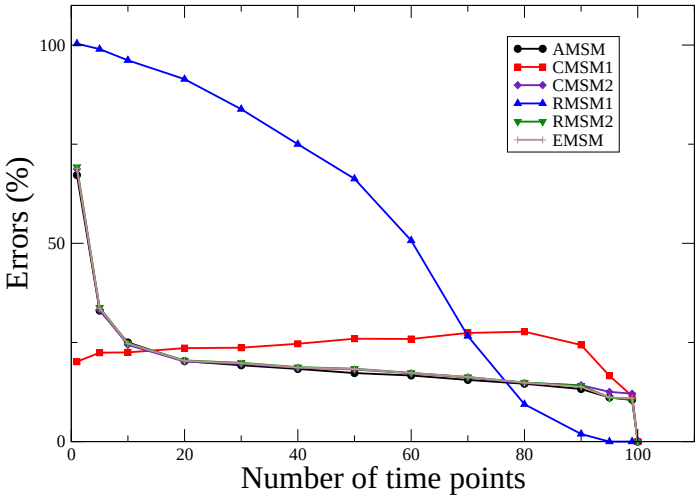


Figure 8.1: Comparison of all models applied to a synthetic network with 100 genes

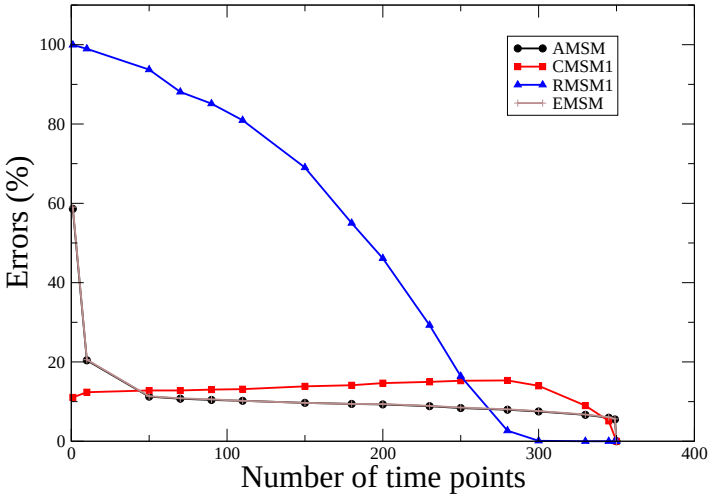


Figure 8.2: Comparison of AMSM, CMSM1, RMSM1, and EMSM applied to a synthetic network with 350 genes

8.1.2 Benchmark subnetworks of *Saccharomyces cerevisiae* and *E. coli*

To evaluate our models on the benchmark subnetworks, we compute the PPV and Se. The PPV measures the ratio of correctly inferred interactions by a model and total number of interactions in the inferred network. The Se is the ratio of correctly inferred interactions by a model over total number of interactions in the real network.

Table 8.1 shows the PPV and Se of our models on 14-gene subnetwork of yeast. The table shows that CMSM1 result is the best overall in terms of those values. In the true network, there are 41 interactions (activation and inhibition) in total out of 196 possible interactions. 9 interactions in signed graph are correctly inferred by CMSM1. Other models correctly infer 6 interactions. As a comparison, Kim *et al.* [56] correctly identified 8 interactions. Although PPV and Se of all models are pretty low for the signed graph, CMSM1 show a bit better result in terms of the number of correctly identified interactions.

The PPV and Se values of all models for the five-gene subnetwork of yeast with different types of data are shown in Table 8.2 and 8.3. First of all, it is a bit surprised that solutions of all models are the same for each data set. Canton *et al.* [15] evaluated several published algorithms using their switch-on and switch-off time series data. We compare our models with two of those algorithms, namely, TSNI (First order ordinary differential equation) [4] and BANJO (Bayesian network) [88] both use the time series data. For the switch-on data, we observe that except the directed graph, our models show better PPV and Se than BANJO. TSNI shows the best results overall. However, the recovered network using the

Table 8.1: PPV and Se values of all models on an yeast cell cycle network

Method	Undirected graph		Directed graph		Signed graph	
	PPV	Se	PPV	Se	PPV	Se
AMSM	0.30	0.32	0.29	0.29	0.17	0.17
CMSM1	0.42	0.47	0.37	0.39	0.21	0.22
CMSM2	0.36	0.38	0.27	0.27	0.12	0.12
RMSM1	0.21	0.21	0.2	0.2	0.13	0.12
RMSM2	0.33	0.38	0.23	0.24	0.14	0.15
LpMSM	0.31	0.38	0.23	0.27	0.13	0.15
EMSM	0.33	0.40	0.21	0.24	0.13	0.15

Table 8.2: PPV and Se values comparison on a five-gene yeast network using switch-on data

Method	Undirected graph		Directed graph		Signed graph	
	PPV	Se	PPV	Se	PPV	Se
TSNI	1	0.57	0.8	0.5	0.6	0.38
BANJO	0.6	0.43	0.3	0.25	0.3	0.25
Our models	0.71	0.71	0.43	0.38	0.43	0.38

switch-off data by our models is much closer to the true network. We identified 4 correct interactions (There are 8 interactions in the true network) as a signed graph. Also, PPV and Se values of signed graph are higher than those of TSNI and BANJO.

The last evaluation of our models using *E. coli* time series data is shown in Table 8.4. ARNACE performs well to recover an undirected graph. Results of our models are close to each other, but CMSM1 performs best for the undirected graph. For the signed graph, PPV and Se are zero, which means that there are no occurrence of TP. Among our models, RMSM1 shows the highest PPV and Se. It identified 19 interactions as the signed graph. (There are 43 interactions in the true network).

The test results are promising, in particular, compared to those reported in the literature. It is expected that the performance of our models depends on the quality of microarray data.

8.2 Future directions

Using the linearized models is only the first step towards building more realistic nonlinear dynamic models. The study of the linearized models not only helps to develop the needed mathematical and numerical capabilities but also provide valuable approximate information about the underlying complicate and nonlinear gene regulatory networks. However, it is well-known fact that complex dynamic networks are rarely described by linear models, and nearly steady state gene expression profiles are difficult to obtain. Our experience also tells that it is not easy to obtain the time-series microarray data near the steady state to validate these linear model. To model the whole gene expression pathways of a genetic network,

Table 8.3: PPV and Se values comparison on a five-gene yeast subnetwork using switch-off data

Method	Undirected graph		Directed graph		Signed graph	
	PPV	Se	PPV	Se	PPV	Se
TSNI	1	0.57	0.8	0.38	0.2	0.13
BANJO	0.8	0.57	0.6	0.38	0.4	0.25
Our models	0.57	0.57	0.5	0.5	0.5	0.5

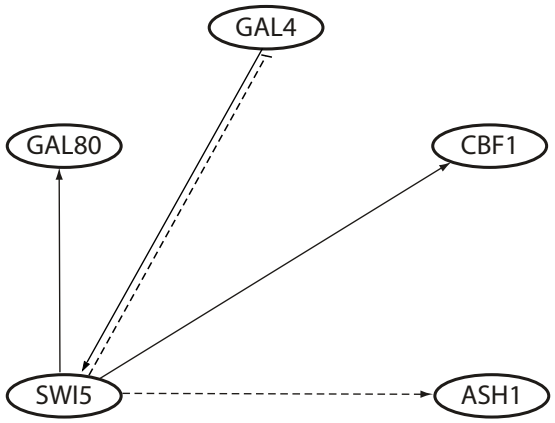


Figure 8.3: Inferred network by TSNI using switch-on data [15]

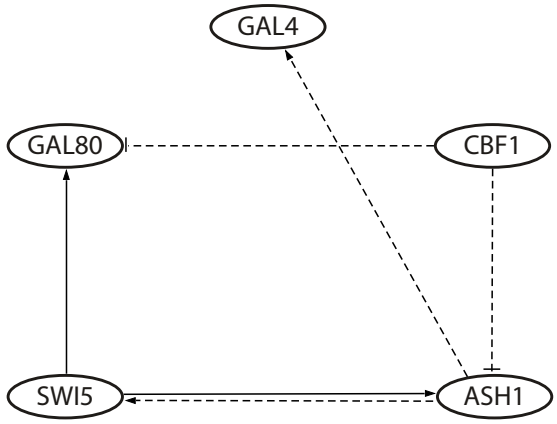


Figure 8.4: Inferred network by BANJO using switch-on data [15]

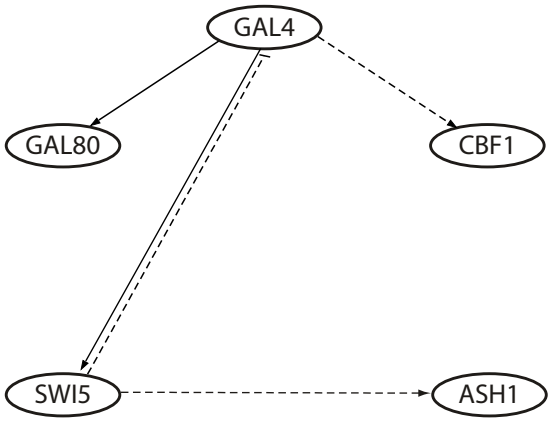


Figure 8.5: Inferred network by TSNI using switch-off data [15]

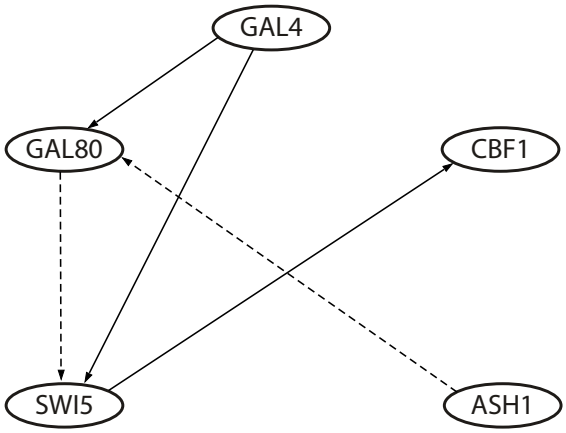


Figure 8.6: Inferred network by BANJO using switch-off data [15]

Table 8.4: PPV and Se values comparison on a fourteen-gene *E.coli.* subnetwork (PPV and Se values of ARANCE, BANJO, Clustering are reported in [3])

Method	Undirected graph		Directed graph		Signed graph	
	PPV	Se	PPV	Se	PPV	Se
ARANCE	0.75	0.37	-	-	-	-
BANJO	0.73	0.69	0.61	0.39	0	0
Clustering	0.9	0.59	-	-	-	-
AMSM	0.68	0.86	0.67	0.65	0.45	0.44
CMSM1	0.68	0.96	0.6	0.65	0.38	0.42
CMSM2	0.73	0.92	0.64	0.68	0.36	0.37
RMSM1	0.66	0.86	0.69	0.72	0.49	0.51
RMSM2	0.7	0.92	0.73	0.74	0.43	0.44
LpMSM	0.71	0.92	0.7	0.72	0.36	0.37
EMSM	0.68	0.88	0.64	0.65	0.36	0.37

we must consider and develop nonlinear differential equation models. As mentioned in Subsection 1.3.1, the crucial first step is to construct/postulate the forms of the influence function $\mathbf{F}$, which clearly requires a lot of biological intuition, knowledge, experiences, and insights, in addition to mathematical modeling experiences, knowledge, and sophistication. Such pre-requisites also make collaboration with biologists necessary and vital in order to be successful.

Another issue to be considered is the quality of experimental data. Currently, it is hard to collect the experimental microarray data which exactly fit with mathematical models. In order to evaluate mathematical models, it needs to design experiments to produce the microarray data that meet the requirements of the models. For example, for the linearized DE models, the microarray data should be collected around steady state point. Also, we need to develop tools such as image processing and statistical methods to pre-process the raw microarray data.

One important and difficult issue has to be faced in various network identification approaches/methods is thresholding. In the context of differential equation modeling, the issue becomes how to post-process the computed network (matrix) obtained by a numerical algorithm which faithfully computes the solution of the underlying mathematical model. Various ad hoc or empirical techniques have been used in the literature. We have proposed a RMT based post-processing technique in Chapter 7. Clearly, the technique is non-trivial and non-intuitive. We plan to further investigate and develop this technique, in particular, lay down a mathematical foundation for the RMT based thresholding technique.

Bibliography

[1] J. Bahler. Cell-cycle control of gene expression in budding and fission yeast. *Annu. Rev. Genet.*, 39:69, 2005.

[2] J. N. Bandyopadhyay and S. Jalan. Universality in complex networks: Random matrix analysis. *Phys. Rev. E*, 76:026109, 2007.

[3] M. Bansal, V. Belcastro, A. Ambesi-Impiombata, and D. di Bernardo. How to infer gene networks from expression profiles. *Molecular Systems Biology*, 3:78, 2007.

[4] M. Bansal, G. Della Gatta, and D. di Bernardo. Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. *Bioinformatics*, 22:815, 2006.

[5] I. Barrodale and F. D. K. Roberts. An improved algorithm for discrete l_1 linear approximation. *SIAM J. Numer. Anal.*, 10:839, 1973.

[6] R. H. Bartels, A. R. Conn, and J. W. Sinclair. Minimizing techniques for piecewise differentiable functions: The l^1 solution to an overdetermined linear system. *SIAM J. Numer. Anal.*, 15:224, 1978.

[7] J. Ben, H. Park, J. Glick, and L. Zhang. Accurate solution to overdetermined linear equations with error using l^1 norm minimization. *Comput. Optim appl.*, 171:737, 2000.

[8] R. Bhatia. *Matrix analysis.* Springer, 1997.

[9] D. W. Boyd. The power method for l^p norms. *Linear algebra and Appl.*, 9:95, 1974.

[10] P. Brazhnik, A. De la Fuente, and P. Mendes. Gene networks: how to put the function in genomics. *Trends Biotechnology*, 20:467, 2002.

[11] R. L. Burden and J. D. Faires. *Numerical Anlysis*. Brooks/Cole, 2001.

[12] R. A. Byrd and D. A. Payne. Convergence of the irls algorithm for robust regression. *Technical reprt, Jonh Hopkins University (Unpublished)*, 313, 1979.

[13] E. J. Candès, J. Romberg, and T. Tao. Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theory*, 52:489, 2006.

[14] E. J. Candès and T. Tao. Decoding by linear programming. *IEEE Trans. Inf. Theory*, 51:4203, 2005.

[15] I. Cantone, L. Marucci, F. Iorio, M. A. Ricci, V. Belcastro, M. Bansal, S. Santini, M. di Bernardo, D. di Bernardo, and M. P. Cosma. A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell*, 137:172, 2009.

[16] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pur suit. *SIAM J. Sci. Comput.*, 1:33, 1998.

[17] T. Chen, H. L. He, and G. M. Church. Modeling gene expression with differential equations. *Pac. Sym. Biocomput.*, 4:29, 1999.

[18] K.-H. Cho, S.-M. Choo, S.H. Jung, J.-R. Kim, H.-S. Choi, and J. Kim. Reverse engineering of gene regulatory networks. *IET Syst. Biol.*, 1:149, 2007.

[19] K.-H. Cho, S.-M. Choo, P. Wellstead, and O. Wolkenhauer. A unified framework for unraveling the functional interaction structure of a biomolecular network based on stimulus-response experimental data. *FEBS Lett.*, 579:4520, 2005.

[20] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, and A. Conway. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell*, 2:65, 1998.

[21] J. F. Claerbout and F. Muir. Robust modeling with erratic data. *Geophysics*, 38:826, 1973.

[22] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Mach. Learn.*, 9:309, 1992.

[23] C. Cosentino, W. Curatola, F. Montefusco, M. Bansal, D. di Bernardo, and F. Amato. Linear matrix inequalities approach to reconstruction of biological networks. *IET Syst. Biol.*, 1:164, 2007.

[24] A.-M. Croicu and M. Y. Hussaini. On the expected optimal value and the optimal expected value. *Appl. Math. Comput.*, 180:330, 2006.

[25] G. Darche. Iterative l^1 deconvolution. *Stanford Exploration Project*, 61:99, 1998.

[26] H. de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *J. Comput. Biol.*, 9:69, 2002.

[27] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[28] D. di Bernardo, T. S. Gardner, and J. J. Collins. Robust identification of large genetic networks. *Proc. Pacific Symp. on Biocompution, Hawaii, USA*, page 486, 2004.

[29] D. di Bernardo, M. Thompson, T. Gardner, S. Chobot, E. Eastwood, A. Wojtovich, S. Elliott, S. Schaus, and J. Collins. Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. *Nat. Biotechnol.*, 23:377, 2005.

[30] D. L. Donoho. Compressed sensing. *IEEE Trans. Inf. Theory*, 52:1289, 2006.

[31] D. L. Donoho and B. F. Logan. Signal recovery and the large sieve. *SIAM J. Appl. Math.*, 52:557, 1998.

[32] D. L. Donoho and P. B. Stark. Uncertainty principles and signal recovery. *SIAM J. Appl. Math.*, 49:906, 1998.

[33] D. L. Donoho and J. Tanner. Sparse nonnegative solutions of underdetermined linear equations by linear programming. *Proc. Nat. Acad. Sci*, 102:9446, 2005.

[34] Sorin Drăghici. *Data analysis tools for DNA microarrays.* Chapman and Hall/CRC, 2003.

[35] F. J. Dyson. Statistical theory of the energy levels of complex systems, I. *J. Math. Phys*, 3:140, 1962.

[36] F. J. Dyson. The threefold way algebraic structure of symmetry groups and ensembles in quantum mechanics. *J. Math. Phys*, 3:1200, 1962.

[37] R. W Farebrother. Unbiased l_1 and l_∞ estimation. *Commun. Statist. Theor. Math.*, 14:1941, 1985.

[38] D. E. Featherstone and K. Broadie. Wrestling with pleiotropy:genomic and topological analysis of the yeast gene expression network. *Bioessays*, 24:267, 2002.

[39] N. Friedman, M. Linial, L. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *J. Comput. Biol.*, 7:601, 2000.

[40] T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301:102, 2003.

[41] T. S. Gardner and J. Farith. Reverse-engineering transcription control networks. *Phys. Life Rev.*, 2:68, 2005.

[42] M. Gaudin. Sur la loi limite de l'espacement des valeurs propres d'une matrice aleatoire. *Nucl. Phys*, 25:447, 1960.

[43] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99:7821, 2002.

[44] I. C. Gohberg and M. G. Krein. *Introduction to the theory of linear nonselfadjoint operators.* Amer. Nath. Soc. Translation, Providence, 1969.

[45] Gene H. Golub and Charles F. Van Loan. *Matrix Computation*. Johns Hopkins University Press, 1996.

[46] A. J. F. Griffiths, W. M. Gelbart, J. H. Miller, and R. C. Lewontin. *Modern Genetic Analysis*. W. H. Freeman and Company, 1999.

[47] T. Guhr, A. Muller-Groeling, and H. A. Weidenmuller. Random-matrix theories in quantum physics: common concepts. *Phys. Rep.*, 299:189, 1998.

[48] G. Hardiman. *Microarrays Methods and Applications*. DNA Press, 2003.

[49] A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. *Pac. Symp. Biocomput.*, page 422, 2001.

[50] N. Higham. Estimating the matrix p-norm. *Numer. Math.*, 62:539, 1992.

[51] N. Higham. *Functions of Matrices: Theory and Computation*. SIAM, 2008.

[52] P. J. Huber. *Robust Statistics*. New York: John Wiley, 1981.

[53] T. Ideker, V. Thorsson, J. A. Ranish, R. Christmas, and J. Buhler. Integrated genomic and protemic analysis of a systematically perturbed metabolic network. *Science*, 292:929, 2001.

[54] S. Imoto, T. Goto, and S. Miyano. Estimation of genetic networks and functional structures between genes by using bayesian networks and nonparametric regression. *Pac. Symp. Biocomput.*, page 175, 2002.

[55] A. Julius, M. Zavlanos, S. Boyd, and G.J. Pappas. Genetic network identification using convex programming. *IET Syst. Biol.*, 3:155, 2009.

[56] J. Kim, D. G. Bates, I. Postlethwaite, P. Heslop-Harrison, and K. Cho. Linear time-varying models can reveal non-linear interactions of biomolecular regulatory networks using multiple time-series data. *Bioinformatics*, 24:1286, 2008.

[57] S. Kim, J. Kim, and K. Cho. Inferring gene regulatory networks from temporal expression profiles under time-delay and noise. *Comp. Biol. and Chem.*, 31:239, 2007.

[58] A. J. Kletywegt and A. Shapiro. *Stochastic optimization, in the Handbook of Industrial Engineering, Garbriel Salvendy ed., pages 2625-2650.* John Wiley, New York, 3rd edition, 2001.

[59] P. E. Kloeden and E. Platen. *Numerical solution of stochastic differential equations, volume 23 of application of mathematics (New York).* Springer-Verlag, Berlin, 1992.

[60] Laloux, P. Cizeau, J. P. Bouchaud, and M. Potters. Noise dressing of financial correlation matrices. *Phys. Rev. Lett*, 83:1467, 1999.

[61] S. Liang, S. Fuhrman, and R. Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Prac. Symp. Biocomout.*, 18:99, 1998.

[62] J. C. Liao, R. Boscolo, Y. L. Yang, L. M. Tran, C. Sabatti, and V. P. Roychowdhury. Network component analysis: reconstruction of regulatory signals in biological systems. *Proc. Natl. Acad. Sci.*, 100:15522, 2003.

[63] F. Lou, J. Zhong, Y.-L. Yang, R. H. Scheuermann, and J. Zhou. Application of random matrix theory of biological networks. *Phys. Lett. A*, 357:420, 2006.

[64] F. Lou, J. Zhong, Y.-L. Yang, and J. Zhou. Application of random matrix theory to microarray data for discovering functional gene modules. *Phys. Rev. E*, 73:031924, 2006.

[65] F. Luo, Y. Yang, J. Zhong, H. Gao, L. Khan, and D. K. Thompson J. Zhou. Constructing gene co-expression networks and predicting functions of unknown genes by random matrix theory. *BMC Bioinformatics*, 8:299, 2007.

[66] F. Luo, J. Zhong, Y. Yang, R. H. Scheuermann, and J. Zhou. Applications of random matrix theory to biological networks. *Phys. Lett. A*, 357:420, 2006.

[67] F. Luo, J. Zhong, Y. Yang, and J. Zhou. Applications of random matrix theory to microarray data for discovering functional gene modules. *Phys. Rev. E*, 73:031924, 2006.

[68] M. Mehta. *Random matrices*. 3rd edition, Elsevier, Amsterdam, 2004.

[69] M. L. Mehta. On the statistical properties of the level-spacing in nuclear spectra. *Nucl. Phys*, 18:395, 1960.

[70] B. Øksendal. *Stochastic differential equations, an introduction with applications*. Springer-Verlag, Berlin, 1985.

[71] R. Porreca, E. Cinquemani, J. Lygeros, and G. Ferrari-Trecate. Identification of genetic network dynamics with unate structure. *Bioinformatics*, 26:1239, 2010.

[72] M. Salzmann, R. Hartley, and P. Fua. Convex optimization for deformable surface 3-d tracking. *IEEE international conference on Computer Vision*, 2007.

[73] F. Santosa and W. W. Symes. Linear inversion of band-limited reflection seismograms. *SIAM J. Stat. Comput.*, 7:1307, 1986.

[74] R. Schatten. *Norm ideals of completely continuous operators*. Springer, Berlin, 1960.

[75] K. Sim and R. Hartley. Recovering camera motion using l_∞ minimization. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[76] B. Simon. *Trace ideals and their applications*. Cambridge University Press, Cambridge, 1979.

[77] E. Sontag, A. Kiyatkin, and B. N. Kholodenko. Inferring dynamic architecture of cellular networks using time series of gene expression, protein and metabolite data. *Bioinformatics*, 20:1877, 2004.

[78] T. Miyano T. Akutsu and S. Kuhara. Algorithms for identifying boolean networks and related networks based on matrix multiplication and fingerprint function. *J. Comput. Biol.*, 7:331, 2000.

[79] H. L. Taylor, S. C. Banks, and J. F. McCoy. Deconvolution with the l1 norm. *Geophysics*, 44:39, 1979.

[80] J. Tegner, M. K. S. Yeung, J. Hasty, and J. J. Collins. Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling. *Proc. of the National Academy of Science*, 100:5944, 2003.

[81] James Watson and Francis Crick. Molecular structure of nucleic acid. *Nature*, 171:737, 1953.

[82] E. P. Wigner. On the statistical distribution of the widths and spacing of nuclear resonance levels. *Proc. Cam. Phil. Soc.*, 47:790, 1951.

[83] E. P. Wigner. Results and theory of resonance absorption, conference on neutron physics by Time-of-flight Gatlinburg Tennessee Nov. 1 and 2, 1956. *Oak Ridge Nat'l Lab report ORNL-2309*, page 59, 1957.

[84] P. C. Winter, G. I. Hickey, and H. L. Fletcher. *Genetics*. BIOS, 2002.

[85] J. Wishart. Generalized product moment distribution in samples. *Biometrika*, 20A:32, 1928.

[86] M. Xiong, J. Li, and X. Fang. Identification of genetic networks. *Genetics*, 166:1037, 2004.

[87] M. K. S. Yeung, J. Hasty, and J. J. Collins. Reverse engineering gene networks using singular value decomposition and robust regression. *Proc. Natl. Acad. Sci. USA*, 99:6163, 2002.

[88] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20:3594, 2004.

[89] L. Zhang, J. Dellinger, and F. Muir. Unique solutions for l^1 problems. *Stanford Exploration Project*, 59:87, 1988.

Appendices

Appendix A

Strictly convex matrix norm

Definition A.0.1. A functional $\mathcal{L} : \mathbf{R}^{n\times n} \to \mathbf{R}$ is called a convex functional, if for any matrices $B, C \in \mathbf{R}^{n\times n}$ there holds

$$\mathcal{L}\big(\alpha B + (1-\alpha)C\big) \leq \alpha\mathcal{L}(B) + (1-\alpha)\mathcal{L}(C) \qquad \forall \alpha \in (0,1). \tag{A.1}$$

Moreover, $\mathcal{L}$ is said to be strictly convex if (A.1) is a true inequality for $B \neq C$.

Definition A.0.2. A matrix norm $\|\cdot\|$ on $\mathbf{R}^{n\times n}$ is called a strictly convex matrix norm, if for any two matrices $B, C \in \mathbf{R}^{n\times n}$ such that $\|B\| = \|C\| = \frac{1}{2}\|B + C\|$ implies $B = C$.

Definition A.0.3. A matrix norm $\|\cdot\|$ on $\mathbf{R}^{n\times n}$ is called a strictly convex matrix norm, if for any two distinct matrices $B, C \in \mathbf{R}^{n\times n}$ such that $\|B\| = \|C\| = 1$ implies $\left\|\frac{B+C}{2}\right\| < 1$.

Proposition 1. Let $\|\cdot\|$ be a matrix norm on $\mathbf{R}^{n\times n}$ and $\mathcal{L}(\cdot) = \|\cdot\|$. The notion of the strict convexity defined by the above three definitions satisfies the following relationships:

(i) Definition A.0.2 and Definition A.0.3 are equivalent.

(ii) Definition A.0.1 is a stronger notion than the other two.

Proof. To show (i), suppose that $\|\cdot\|$ is strictly convex in the sense of Definition A.0.2, then for any two matrices $B, C \in \mathbf{R}^{n\times n}$ with $\|B\| = \|C\| = 1$ and $\|B + C\| = 2$, there must hold $B = C$. Equivalently, for any two distinct matrices $B, C \in \mathbf{R}^{n\times n}$ with $\|B\| = \|C\| = 1$

there must hold $\|\frac{1}{2}(B+C)\| < 1$. Hence, $\|\cdot\|$ is strictly convex in the sense of Definition A.0.3.

Next, suppose that $\|\cdot\|$ is strictly convex in the sense of Definition A.0.3, then for any two distinct matrices $B, C \in \mathbf{R}^{n\times n}$ with $\|B\| = \|C\| = 1$ there holds $\left\|\frac{1}{2}(B+C)\right\| < 1$ or $\|B+C\| < 2$. Suppose $\|B\| = \|C\| = \frac{1}{2}\|B+C\|$. Clearly, if $\|B\| = \|C\| = 0$ then $B = C = 0$. Now suppose $\|B\| = \|C\| \neq 0$, set $B' = \frac{B}{\|B\|}$, $C' = \frac{C}{\|C\|}$, then $\|B'\| = \|C'\| = 1$. If $B \neq C$, then $B' \neq C'$. But $1 = \|B'\| = \|C'\| = \frac{1}{2}\|B' + C'\|$ contradicts with the convexity assumption (in the sense of Definition A.0.3). Hence, $B' = C'$, or $B = C$. Therefore, $\|\cdot\|$ is strictly convex in the sense of Definition A.0.2.

(ii) can be proved using a counterexample. Frobineous norm is strictly convex in the sense of Definition A.0.2 (cf. Chapter 3). We now prove that every matrix norm is not strictly convex in the sense of Definition A.0.1. We choose two distinct matrices B and $C = mB$ for a nonzero constant m. Let $\mathcal{L}$ be any norm, then

$$\begin{aligned}
\mathcal{L}(\alpha B + (1-\alpha)C) =& \mathcal{L}(\alpha B + (1-\alpha)mB) \\
=& \mathcal{L}((\alpha + (1-\alpha)m)B) \\
=& |\alpha + (1-\alpha)m|\mathcal{L}(B) \quad \text{(since } \mathcal{L} \text{ is a norm)} \\
=& (\alpha + (1-\alpha)m)\mathcal{L}(B). \quad (\alpha, 1-\alpha, m \text{ are nonnegative})
\end{aligned}$$

The right-hand side of (A.1) is

$$\begin{aligned}
\alpha\mathcal{L}(B) + (1-\alpha)\mathcal{L}(C) &= \alpha\mathcal{L}(B) + (1-\alpha)\mathcal{L}(mB) \quad \text{(since } \mathcal{L} \text{ is a norm)} \\
&= \alpha\mathcal{L}(B) + (1-\alpha)m\mathcal{L}(B) \\
&= (\alpha + (1-\alpha)m)\mathcal{L}(B).
\end{aligned}$$

Thus, (A.1) holds for two distinct matrices B and C, this then proves the claim that all matrix norms are not strictly convex in the sense of Definition A.0.1. Therefore, the strict convexity of Definition A.0.1 is a stronger notion than that of the other two definitions. □

For a general functional, Definition A.0.1 is often used to verify the strict convexity. Definition A.0.2 and Definition A.0.3 are used for special functionals which are matrix norms or vector norms.

Singular value decomposition (SVD)

Theorem A.0.1. (*Singular value decomposition*)
For any given matrix M $\in \mathbf{R}^{m\times n}$ with $m < n$, there exists a decomposition (see [27] for a proof)

$$\mathrm{M} = \mathrm{U}\Lambda\mathrm{V}^T,$$

where U $\in \mathrm{R}^{m\times m}$ and V $\in \mathrm{R}^{n\times n}$ are orthogonal matrices and $\Lambda \in \mathrm{R}^{m\times n}$ is non-negative diagonal matrix. The decomposition is called the singular value decomposition of M.

Remark A.0.1. (*Properties of SVD*)[27, 51, 45]

(i) A SVD exists for any (real or complex) matrix of any size (square or rectangular).

(ii) Diagonal values of Λ are called the singular values of M.

(iii) The diagonal values of Λ are $\sigma_i = \{\sqrt{\lambda_i}\}$ where $\{\lambda_i\}_{i=1}^m$ are the eigenvalues of M^TM and MM^T.

Assume that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_m = 0$, then

(iv) rank$(M) = r$: number of nonzero singular values.

(v) Columns of U corresponding to non-zero entries of M span the range of M, i.e. $range(M) = span\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_r\}$.

(vi) Columns of V corresponding to zero entries of M span the null-space of M, i.e. $null(M) = span\{\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \cdots, \mathbf{v}_n\}$.

Unitary invariant norm

Definition A.0.4. (*Unitary invariant norm*)[8]

A matrix norm $\|\cdot\|$ is called an unitary invariant norm, if for any matrix $A \in \mathbf{R}^{n\times m}$ (or $\mathbf{C}^{n\times m}$) there holds $\|A\| = \|UAV\|$ for all unitary matrices U, V.

Appendix B

The following codes are written in Matlab®. All the m-files for the numerical tests are listed below. The function `pnorm` used in LpMSM is from [50].

Table B.1: List of m-files

Function name	Description
`GRN`	Compute solutions of all models.
`CMSM1`	Compute a CMSM1 solution.
`CMSM2`	Compute a CMSM2 solution.
`RMSM1`	Compute a RMSM1 solution.
`RMSM2`	Compute a RMSM2 solution.
`EMSM`	Compute a EMSM solution.
`LpMSM`	Compute a LpMSM solution.
`Infer`	Produce a post-processed regulatory matrix.
`PS`	Compute PPV and Se values for undirected, directed, and signed graphs.
`NNSD`	Plot the NNSD of eigenvalues of matrix.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [A_AMSM,A_CMSM1,A_CMSM2,A_RMSM1,
          A_RMSM2,A_EMSM,A_LpMSM]=GRN(X,Y,C,N,q)
% GRN computes a solution of min||A|| subject to AX=Y
% using all models.
%% Input
% X : n by m Time-series microarray data matrix
%    (n: number of genes, m: number of time points).
% Y : n by m rate of changes of time-series matrix.
% C : n by (n-m) initial guess matrix for models that use the
%     gradient descent method.
% N : number of iteration for gradient descent method.
% q : scalar for EMSM (entrywise p norm)and LpMSM (matrix p norm).
%%Output
% Solutions of all models.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% AMSM solution
Z=X/(X'*X);
A_AMSM=Y*Z';
% Singular value decomposition of X'
[U,S,V]=svd(X');
[n,m]=size(X);
p=rank(X');
s=n-p;
V0=V(:,p+1:n)
% CMSM1 solution
A_CMSM1=CMSM1(A_AMSM,V0,p);
% CMSM2 solution
```

```
A_CMSM2=CMSM2(A_AMSM,V0,C,N);

% RMSM1 solution

A_RMSM1=RMSM1(A_AMSM,X,V0,n,s,p);

% RMSM2 solution

A_RMSM2=RMSM2(A_AMSM,V0,C,N);

% EMSM solution

A_EMSM=EMSM(A_AMSM,V0,C,N,q);

% LpMSM solution

A_LpMSM=LpMSM(A_AMSM,V0,C,N,q);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A_CMSM1 = CMSM1(A,B,p)
% CMSM1 computes a solution of CMSM1 by computing
% C0 of min||A'-C0'*B||Linfinity column-by column using linear
% programming.
%% Input
% A: Particular solution.
% B: V0=V(:,p+1:n) where V is from [U,S,V]=svd(X').
% p: rank of X'.
%%Output
% A_CMSM1: CMSM1 solution.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[n,n]=size(A);
% initialize for C0
C0=zeros(n-p,n);
% Compute each column of C0 using LP function.
% Take each column of A for the right hand side vector of linear
% system.
% B is coefficient matrix.
for i=1:n
   b=A(i,:)';
   C0(:,i)=LP(B,b);
end
% Assemble CMSM1 solution C0'*V2' is the general solution
A_CMSM1=A-C0'*V0';


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x_cheb=LP(A,b)
```

```
% LP computes a solution of minimizing residual of linear system
% in the vector l infinity  norm using linear programming method.
%%Input
% A : Coefficient matrix of linear system.
% b : Right hand side vector of linear system.
%% Output
% x_cheb : Solution vector of l infinity minimization.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Convert a linear system to linear programming problem.
[m,n]=size(A);
f = [ zeros(n,1); 1 ];
Ane = [ +A, -ones(m,1) ; ...
    -A, -ones(m,1) ];
 bne = [ +b; -b ];
 xt = linprog(f,Ane,bne);
x_cheb = xt(1:n,:);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A_CMSM2=CMSM2(A,B,C,N)
% CMSM2 computes a solution of CMSM2 by computing
% C0 of min||A-C0*B'||L1 using gradient descent
% method.
%% Input
% A: Particular solution.
% B: V0=V(:,p+1:n) where V is from [U,S,V]=svd(X').
% C: Initial guess of C0.
% N: Number of maximum iterations.
%%Output
% A_CMSM2: CMSM2 solution.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 TolFun = 1e-8;   %|f(x)| < TolFun wanted
 TolX = 1e-6;     %|x(k)- x(k - 1)|<TolX wanted
% Gradient Descent method
x=C;
fx0 = norm(A-x*B',1);
for k=1:N
    z=grad_L1(A,B,x);
    z=z/norm(z,1);
    a=alpha_L1(A,B,x);
  if a==0
    break
  else
    x=x-a*z;
    fx = norm(A-x*B',1);
  end
```

```
    if(norm(x - x_0,1) < TolX & abs(fx - fx0) < TolFun),
       fprintf(1, 'k = %d\n',k);
       break;
    end
    x_0 = x;
    fx0 = fx;
    fprintf(1, 'k = %d\n',k);
end
if k == N,
    fprintf(1,'Just best in %d iterations:',N),
end
% Assemble CMSM2 solution.
 A_CMSM2=A-x*B';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function GradA=grad_L1(A,B,C)
% grad_L1 computes gradient of F(C)=||A-C*B'||_L1 using finite
% difference approximation.
%% Input
% A: Particular solution.
% B: V0=V(:,p+1:n) where V is from [U,S,V]=svd(X').
%%Output
% GradA: gradient of ||A-C*B'||_L1.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[n,m]=size(C);
GradA=zeros(n,m);
h=0.01;
for i=1:n
```

```
    for j=1:m
        LC=C; RC=C;
        LC(i,j)=C(i,j)+h;
        RC(i,j)=C(i,j)-h;
        GradA(i,j)=(norm(A-LC*B',1)-norm(A-RC*B',1))/2*h;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=alpha_L1(A,B,C)
% alpha_L1 calculates a decreasing step length alpha for the
% gradient descent technique for minimizing functions.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g_1=norm(A-C*B',1);
z=grad_L1(A,B,C);
if norm(z,1)==0
  y=0;
else
  z=z/norm(z,1);
  alpha_1=0;
  alpha_3=1;
  g_3=norm(A-(C-alpha_3*z)*B',1);
  while g_3>=g_1
    alpha_3=(alpha_3)/2;
    g_3=norm(A-(C-alpha_3*z)*B',1);
  end
  alpha_2=alpha_3/2;
  g_2=norm(A-(C-alpha_3*z)*B',1);
```

```
  h_1=(g_2-g_1)/alpha_2;
  h_2=(g_3-g_2)/(alpha_3-alpha_2);
  h_3=(h_2-h_1)/alpha_3;
  alpha_0=0.5*(alpha_2-(h_1/h_3));
  g_0=norm(A-(C-alpha_3*z)*B',1);
  if min(g_0,g_3)==g_0
    y=alpha_0;
  end
  if min(g_0,g_3)==g_3
    y=alpha_3;
  end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A_RMSM1 = RMSM1(A,B,p)
% RMSM1 computes a solution of RMSM1 by computing
% C0 of min||A'-C0'*B||L1 column-by column using IRLS.
%% Input
% A: Particular solution.
% B: V0=V(:,p+1:n) where V is from [U,S,V]=svd(X').
% p: rank of X'.
%%Output
% A_RMSM1: RMSM1 solution.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[n,n]=size(A1);
eps=1e-5;
tol1=1e-3;
maxit=100;
% Initializing C0.
C0=zeros(n-p,n);
% Compute each column of C0 using IRLS function.
% Take each column of A for the right hand side vector of linear system.
% B is coefficient matrix of linear system.
for i=1:n
    b=A(i,:)';
    C0(:,i)=IRLS(B,b,eps,tol1,maxit,1);
end
% C2'*V2' is the general solution.
A_RMSM1=A_1-C)'*B';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function z=IRLS(A,b,eps,tol1,maxit,p)
% IRLS computes a solution of minimizing residual of linear system
% in the vector l1 norm using IRLS method.
%%Input
% A : Coefficient matrix of linear system.
% b : Right hand side vector of linear system.
%% Output
% x_cheb : Solution vector of l1 minimization.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[n,m]=size(B)
% Initial guess for the first iteration
W=eye(n,n);
k=0;
while k <= maxit
    k=k+1;
    % compute vector z of B'*W*b*z=B'*W*b
    z=(B'*W*B)\(B'*W*b);
    % compute new W with previous E
    w=b-B*z;
    for i=1:n
        if abs(w(i)) > eps
            w(i)=abs(w(i))^(p-2);
        else
            w(i)=eps^(p-2);
        end
    end
    W=diag(w);
    % check the residual
```

```
    residual=B'*W*B*z-B'*W*b;
    if norm(residual) < tol1
       % fprintf(1, 'k = %d\n',k);
        break
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A_RMSM2=RMSM2(A,B,C,N)
% RMSM2 computes a solution of RMSM2 by computing
% C0 of min||A-C0*B'||Linfinity using gradient descent method.
%% Input
% A: Particular solution.
% B: V0=V(:,p+1:n) where V is from [U,S,V]=svd(X').
% C: Initial guess of C0.
% N: Number of maximum iterations.
%%Output
% A_RMSM2: RMSM2 solution.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TolFun = 1e-5; %|f(x)| < TolFun wanted
TolX = 1e-5;   %|x(k)- x(k - 1)|<TolX wanted
% Gradient descent method.
x=C;
fx0 = norm(A-x*B',inf);
for k=1:N
    z=grad_Linf(A,B,x);
    z=z/norm(z,inf);
    a=alpha_Linf(A,B,x);
  if a==0
    break
  else
    x=x-a*z;
    fx = norm(A-x*B',inf);
  end
    if(norm(x - x_0,inf) < TolX & abs(fx - fx0) < TolFun),
```

```
      % fprintf(1, 'k = %d\n',k);
       break;
    end
    x_0 = x;
    fx0 = fx;
    fprintf(1, 'k = %d\n',k);
end
if k == N,
    fprintf(1,'Just best in %d iterations:',N),
end
% Assemble RMSM2 solution.
A_RMSM2=A-x*B';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function GradA=grad_Linf(A,B,C)
% grad_Linf computes gradient of F(C)=||A-C*B'||_Linfinity using finite
% difference approximation.
%% Input
% A: Particular solution.
% B: V0=V(:,p+1:n) where V is from [U,S,V]=svd(X').
%%Output
% GradA: gradient of ||A-C*B'||_Linfinity at C.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GradA=zeros(n,m);
h=0.01;
for i=1:n
    for j=1:m
        LC=C; RC=C;
```

```
        LC(i,j)=C(i,j)+h;
        RC(i,j)=C(i,j)-h;
        GradA(i,j)=(norm(A-LC*B',inf)-norm(A-RC*B',inf))/2*h;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=alpha_Linf(A,B,C)
% alpha_Linfinity calculates a decreasing step length alpha for the
% gradient descent technique for minimizing functions.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g_1=norm(A-C*B',inf);
z=grad_Linf(A,B,C);
if norm(z,inf)==0
  y=0;
else
  z=z/norm(z,inf);
  alpha_1=0;
  alpha_3=1;
  g_3=norm(A-(C-alpha_3*z)*B',inf);
  while g_3>=g_1
    alpha_3=(alpha_3)/2;
    g_3=norm(A-(C-alpha_3*z)*B',inf);
  end
  alpha_2=alpha_3/2;
  g_2=norm(A-(C-alpha_3*z)*B',inf);
  h_1=(g_2-g_1)/alpha_2;
  h_2=(g_3-g_2)/(alpha_3-alpha_2);
```

```
  h_3=(h_2-h_1)/alpha_3;
  alpha_0=0.5*(alpha_2-(h_1/h_3));
  g_0=norm(A-(C-alpha_3*z)*B',inf);
  if min(g_0,g_3)==g_0
    y=alpha_0;
  end
  if min(g_0,g_3)==g_3
    y=alpha_3;
  end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A_EMSM=EMSM(A,B,C,N,p)
% EMSM computes a solution of EMSM by computing
% C0 of min(||A-C0*B'||ep)^p using gradient descent method.
%% Input
% A: Particular solution.
% B: V0=V(:,p+1:n) where V is from [U,S,V]=svd(X').
% C: Initial guess of C0.
% N: Number of maximum iterations.
% p: Scalar of entrywise p norm.
%%Output
% A_EMSM: EMSM solution.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TolFun = 1e-3;  %|f(x)| < TolFun wanted
TolX = 1e-6;    %|x(k)- x(k - 1)|<TolX wanted
% Gradient descent method.
x=C;
fx0 = elm_norm(A-x*B',p);
for k=1:N
    z=grad_elm(A,B,x,p);
   z=z/elm_norm(z,p);
   a=alpha_elm(A,B,x,p);
  if a==0
    break
  else
    x=x-a*z;
    fx = elm_norm(A-x*B',p);
  end
```

```
    if (norm(x - x_0,'fro') < TolX & abs(fx - fx0) < TolFun)
       fprintf(1, 'k = %d\n',k);
       break;
    end
    x_0 = x ;
    fx0 = fx;
end
if k == N,
    fprintf(1,'Just best in %d iterations:',N),
end
% Assemble A_EMSM solution.
A_EMSM=A-x*B';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a=elm_norm(A,p)
% elm_norm computes the pth power of entrywise p norm of matrix.
%% Input
% A: n by m matrix.
%% Output
% a: pth power of entrywise p norm of matrix A.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a=(sum(sum((abs(A).^q))));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function GradA=grad_elm(A,B,C,p)
% grad_elm computes gradient of F(C)=(||A-C*B'||_elm)^p.
%% Input
% A: Particular solution.
```

```
% B: V0=V(:,p+1:n) where V is from [U,S,V]=svd(X').
%%Output
% GradA: gradient of (||A-C*B'||_elm)^p at C.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M=A-B*C';
S=M./abs(M);
if mod(p,2)==0
    D=-p*M.^(p-1)*C;
else
    D=-p*S.*(M.^(p-1))*C;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=alpha_elm(A,B,C,p)
% alpha_elm calculates a decreasing step length alpha for the
% gradient descent technique for minimizing functions.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g_1=elm_norm(A-C*B',p);
z=grad_elm(A,B,C,p);
if elm_norm(z,p)==0
  y=0;
else
  z=z/elm_norm(z,p);
  alpha_1=0;
  alpha_3=10;
  g_3=elm_norm(A-(C-alpha_3*z)*B',p);
  %%%% error part %%%%
  while g_3>=g_1
```

```
    alpha_3=(alpha_3)/2;
    g_3=elm_norm(A-(C-alpha_3*z)*B',p);
  end
  %%%%%%%%%%%%%%%%%%%%%%%%
  alpha_2=alpha_3/2;
  g_2=elm_norm(A-(C-alpha_3*z)*B',p);
  h_1=(g_2-g_1)/alpha_2;
  h_2=(g_3-g_2)/(alpha_3-alpha_2);
  h_3=(h_2-h_1)/alpha_3;
  alpha_0=0.5*(alpha_2-(h_1/h_3));
  g_0=elm_norm(A-(C-alpha_3*z)*B',p);
  if min(g_0,g_3)==g_0
    y=alpha_0;
  end
  if min(g_0,g_3)==g_3
    y=alpha_3;
  end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A_LpMSM=LpMSM(A,B,C,N,p)
% EMSM computes a solution of RMSM2 by computing
% C0 of min(||A-C0*B'||Lp)^p using gradient descent method.
%% Input
% A: Particular solution.
% B: V0=V(:,p+1:n) where V is from [U,S,V]=svd(X').
% C: Initial guess of C0.
% N: Number of maximum iterations.
% p: Scalar of matrix p norm.
%%Output
% A_LpMSM: LpMSM solution.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TolFun = 1e-3;%|f(x)| < TolFun wanted
TolX = 1e-6;    %|x(k)- x(k - 1)|<TolX wanted
% Gradient descent meothod.
x=C;
fx0 = (pnorm(A-x*B',p))^p;
for k=1:N
    z=grad_Lp(A,B,x,p);
    z=z/pnorm(z,p);
    a=alpha_Lp(A,B,x,p);
  if a==0
    break
  else
    x=x-a*z;
    fx = (pnorm(A-x*B',p))^p;
  end
```

```
    if (norm(x - x_0,'fro') < TolX &abs(fx - fx0) < TolFun)
      % fprintf(1, 'k = %d\n',k);
       break;
    end
    x_0 = x ;
    fx0 = fx;
end
if k == N,
    fprintf(1,'Just best in %d iterations:',N),
end
% Assemble A_LpMSM solution.
A_LpMSM=A-x*B';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function est=pnorm(A, p, tol, noprint)
% pnorm computes matrix p norm.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin < 2, error('must specify norm via second parameter'), end
[m,n]=size(A);
if min(m,n) == 1, est= norm(A,p); return, end
if nargin < 4, noprint = 0;end
if nargin < 3, tol = 1e-4; end
% Stage I. Use algorithm OSE to get starting vector for power method.
%Form y=B*x, at each stage choosing x(k)=c and scaling previous
%x(k+1:n) by s, where norm([c,s],p)=1.
sm =9; % Number of samples.
y=zeros(m,1); x=zeros(n,1);
```

```
for k=1:n
    if k == 1
        c =1; s=0;
    else W= [A(:,k) y];
        if p == 2 % special case. Solve exactly for 2-norm.
            [UI,S,V]=svd(full(W));
            c = V(1,1); s=V(2,1);
        else
            fopt = 0;
            for th=seqa(0,pi,sm)
                c1 = cos(th); s1 = sin(th);
                nrm = norm ([c1 s1],p);
                c1=c1/nrm; s1=s1/nrm;
                f = norm(W*[c1 s1]', p );
                if f > fopt
                    fopt = f;
                    c = c1; s = s1;
                end
            end
        end
    end
    x(k)=c;
    y=x(k)*A(:,k)+s*y;
    if k > 1, x(1:k-1)=s*x(1:k-1); end
end
est = norm(y,p);
if noprint, fprint('Alg OSE: %9.4e\n', est), end
%Stage II. Apply algorithm PM(the power method).
```

```
q = dual(p);
k=1;
while 1
    y=A*x;
    est_old=est;
    est = norm(y,p);
    z=A'*dual(y,p);
    if noprint
        fprint('%2.0f: norm(y) =  %9.4e, norm(z) = %9.4e',k, norm(y,p), norm(z,q)')
        fprint{' real_incr(est) = %9.4e\n', (est-est_old)/est}
    end
    if (norm(z,q) <= z'*x | abs(est-est_old)/est <= tol) & k > 1
        return
    end
    x=dual(z,q);
    k=k+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = dual(x, p)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if max(size(x)) == 1 & nargin == 1
    p=x;
end
if p == 1
    q = inf;
else
    q=1/(1-1/p);
```

```
end
if max(size(x)) == 1 & nargin == 1
    y=q;
    return
end
if norm(x,inf) == 0, y = x; return, end
if p == 1
    y = sign(x)+(x == 0);
elseif p ==inf
    [xmax, k]= max(abs(x));
    f = find(abs(x)==xmax); k=f(1);
    y=zeros(size(x));
    y(k)=sign(x(k));
else
        x=x/norm(x,inf);
    y=abs(x).^(p-1) .*(sign(x) + (x==0) );
    y = y/norm(y,q);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = seqa(a, b, n)
%SEQA   Generate an additive sequence.
%       Y = SEQA(A, B, N) produces a row vector comprising N equally
%       spaced numbers starting at A and finishing at B.
%       If N is omitted then 10 points are generated.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin == 2, n = 10; end
if n <= 1
```

```
   y = a;
   return
end
y = [a+(0:n-2)*(b-a)/(n-1), b];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function GradA=grad_Lp(A,B,C,p)
 grad_Lp computes gradient of F(C)=(||A-C*B'||_Lp)^p.
%% Input
% A: Particular solution.
% B: V0=V(:,p+1:n) where V is from [U,S,V]=svd(X').
%%Output
% GradA: gradient of (||A-C*B'||_Lp)^p at C.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[n,m]=size(C);
GradA=zeros(n,m);
h=0.01;
for i=1:n
    for j=1:m
        LC=C; RC=C;
        LC(i,j)=C(i,j)+h;
        RC(i,j)=C(i,j)-h;
        GradA(i,j)=((pnorm(A-LC*B',p))^p-(pnorm(A-RC*B',p))^p)/2*h;
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=alpha_Lp(A,B,C,p)
% alpha_Lp calculates a decreasing step length alpha for the
% gradient descent technique for minimizing functions.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g_1=(pnorm(A-C*B',p))^p;
z=grad_Lp(A,B,C,p);
if pnorm(z,p)==0
  y=0;
else
  z=z/pnorm(z,p);
  alpha_1=0;
  alpha_3=10;
  g_3=(pnorm(A-(C-alpha_3*z)*B',p))^p;
  %%%% error part %%%%
  while g_3>=g_1
    alpha_3=(alpha_3)/2;
    g_3=(pnorm(A-(C-alpha_3*z)*B',p))^p;
  end
  %%%%%%%%%%%%%%%%%%%%%%%%
  alpha_2=alpha_3/2;
  g_2=(pnorm(A-(C-alpha_3*z)*B',p))^p;
  h_1=(g_2-g_1)/alpha_2;
  h_2=(g_3-g_2)/(alpha_3-alpha_2);
  h_3=(h_2-h_1)/alpha_3;
  alpha_0=0.5*(alpha_2-(h_1/h_3));
  g_0=(pnorm(A-(C-alpha_3*z)*B',p))^p;
  if min(g_0,g_3)==g_0
```

```
    y=alpha_0;
  end
  if min(g_0,g_3)==g_3
    y=alpha_3;
  end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function B=infer(A,r)
% infer produces the post-processed regulatory network by setting
% elements of matrix that are less than a threshold value.
%% Input
% A: Result matrix from each model.
% x: Magnitude value.
%% Output
% B: Post-processed matrix.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[n,m]=size(A);
M=max(max(A));
% Normalize matrix
A=A/M;
% Find a threshold value
e=r*min(min(abs(A)));
for i=1:c
    for j=1:d
        %remove self connection
        if i==j
           A(i,j)= 0;
        elseif abs(A(i,j)) < e
            A(i,j)=0;
        end
    end
end
B=A;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [uPPV,uSe, dPPV, dSe,sPPV,sSe,r,p]=PS(A,B);
% PS computes the PPV and Se of inferred network.
%% Input
% A: Real network(signed network).
% B:Inferred network.
%% Output
% uPPV: PPV value of undirected graph.
% uSe: Se value of undirected graph.
% dPPV: PPV value of directed graph.
% dSe: Se value of directed graph.
% sPPV: PPV value of signed graph.
% sSe: Se value of signed graph.
% r: number of nonzero elements in the real network.
% p: number of nonzero elements in the post-processed
%    inferred network.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[n,m]=size(A);
% Initialize undirected (U), directed(D), signed(S) graphs of
% real network and inferred network .
UA=zeros(n,m); DA=zeros(n,m); SA=zeros(n,m);
UB=zeros(n,m); DB=zeros(n,m); SB=zeros(n,m);
% Post-process an inferred network
B=infer(C);
% Counting number of nonzero elements in the real network and
% the post-processed inferred network.
r=0;p=0
for i=1:n
```

```
    for j=1:n
        if A(i,j)~=0
            r=r+1;
        end
        if B(i,j)~=0
            p=p+1;
        end
    end
end
for i=1:n
    for j=1:m
        % Convert real network to undirected, directed, Signed network
        if A(i,j)~= 0
            UA(i,j)=1;
            UA(j,i)=1;
            DA(i,j)=1;
            if A(i,j)>0
                SA(i,j)=1;
            else
                SA(i,j)=-1;
            end
        end
         % Convert inferred network to undirected, directed, Signed network
        if B(i,j)~= 0
            UB(i,j) = 1;
            UB(j,i)=1;
            DB(i,j)=1;
            if B(i,j)>0
```

```
                SB(i,j)=1;
            else
                SB(i,j)=-1;
            end
        end
    end
end
% Count number of TP, FP, and FN for each type of graphs
uTP=0;uFP=0;uFN=0;dTP=0;dFP=0;dFN=0;sTP=0;sFP=0;sFN=0;
for i=1:n
    for j=1:m
        if UA(i,j)==1 & UB(i,j)==1
            uTP=1+uTP;
        elseif UA(i,j)==0 & UB(i,j)==1
            uFP=1+uFP;
        elseif UA(i,j)==1 & UB(i,j)==0
            uFN=1+uFN;
        end
        if DA(i,j)==1 & DB(i,j)==1
            dTP=1+dTP;
        elseif DA(i,j)==0 & DB(i,j)==1
            dFP=1+dFP;
        elseif DA(i,j)==1 & DB(i,j)==0
            dFN=1+dFN;
        end
        if SA(i,j)~=0 & SA(i,j)==SB(i,j)
            sTP=1+sTP;
        elseif SB(i,j)~= 0 & SA(i,j)~=SB(i,j)
```

```
            sFP=1+sFP;
        end
        if SA(i,j)~= 0 & SA(i,j) ~= SB(i,j)
            sFN=1+sFN;
        end
    end
end
% Calculate PPV and Se values of each type of graphs
uPPV=uTP/(uTP+uFP);
uSe=uTP/(uTP+uFN);
dPPV=dTP/(dTP+dFP);
dSe=dTP/(dTP+dFN);
sPPV=sTP/(sTP+sFP);
sSe=sTP/(sTP+sFN);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function NNSD(A)
% NNSD plots the NNSD of singular values of matrix A.
%% Input
% A : n times n matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[n,m]=size(A);
% Nationalizing spacing vector
S=zeros(n-1,1);
E=sort(svd(A));
Y=[1:n]';
N=bar(E,Y);
hold on
X=E(1:n-10);
YY=Y(1:n-10);
P=polyfit(X,YY,1);
e=polyval(P,E);
plot(E,e)
for i=1:n-1
    S(i)=e(i+1)-e(i);
end
ds=max(S)/sqrt(n);
XX=[min(S):ds:max(S)];
hist(S,XX);
n_elements= histc(abs(S),XX);
c_elements = n_elements/(ds*(n-1));
hold on
w=[0:0.01:4];
```

```
WD=pi*w.*exp(-pi*w.^2/4)./2;
plot(w,WD,'b')
WX=pi*XX.*exp(-pi*XX.^2/4)./2;
ERR=norm(c_elements'-WX,'fro');
xlabel('s')
ylabel('P(s)')
title('NNSD of eigenvalues')
```

Vita

Miun Yoon was born in Busan, Korea (Republic of) on February 6, 1980. After completing high school at Busan Jin girl's high school in 1998, she attended Dong-A University in Korea (March 1998 - February 2003), where she received her Bachelor of Science degree in Mathematics. During that period, she studied English in Canada for one year. She then entered the University of Tennessee at Knoxville in the fall of 2003. In December, 2006, she graduated from the University of Tennessee, Knoxville with a Master of Science degree in Mathematics.