Parameter optimization of linear ordinary differential equations with application in gene regulatory network inference problems

YUE DENG

Master of Science Thesis
Stockholm, Sweden 2014

Parameter optimization of linear ordinary differential equations with application in gene regulatory network inference problems

YUE DENG

Master's Thesis in Scientific Computing (30 ECTS credits)
Master Programme in Computer simulation for Science and Engineering (120 credits)
Royal Institute of Technology year 2014
Supervisors at Unit of Computational Medicine, Karolinska Institutet, Sweden, were Narsis Kiani and Hector Zenil
Examiner was Michael Hanke

TRITA-MAT-E 2014:60
ISRN-KTH/MAT/E--14/60--SE

Royal Institute of Technology
School of Engineering Sciences

KTH SCI
SE-100 44 Stockholm, Sweden

URL: www.kth.se/sci

Abstract

In this thesis we analyze parameter optimization problems governed by linear ordinary differential equations (ODEs) and develop computationally efficient numerical methods for their solution. In addition, a series of noise-robust finite difference formulas are given for the estimation of the derivatives in the ODEs. The suggested methods have been employed to identify Gene Regulatory Networks (GRNs).

GRNs are responsible for the expression of thousands of genes in any given developmental process. Network inference deals with deciphering the complex interplay of genes in order to characterize the cellular state directly from experimental data. Even though a plethora of methods using diverse conceptual ideas has been developed, a reliable network reconstruction remains challenging. This is due to several reasons, including the huge number of possible topologies, high level of noise, and the complexity of gene regulation at different levels. A promising approach is dynamic modeling using differential equations. In this thesis we present such an approach to infer quantitative dynamic models from biological data which addresses inherent weaknesses in the current state-of-the-art methods for data-driven reconstruction of GRNs. The method is computationally cheap such that the size of the network (model complexity) is no longer a main concern with respect to the computational cost but due to data limitations; the challenge is a huge number of possible topologies. Therefore we embed a filtration step into the method to reduce the number of free parameters before simulating dynamical behavior. The latter is used to produce more information about the network's structure.

We evaluate our method on simulated data, and study its performance with respect to data set size and levels of noise on a 1565-gene *E.coli* gene regulatory network. We show the computation time over various network sizes and estimate the order of computational complexity. Results on five networks in the benchmark collection DREAM4 Challenge are also presented. Results on five networks in the benchmark collection DREAM4 Challenge are also presented and show our method to outperform the current state of the art methods on synthetic data and allows the reconstruction of bio-physically accurate dynamic models from noisy data.

Keywords— ordinary differential equations, parameter optimization, gene regulatory network inference, DREAM4 project

Referat

Parameteroptimering av linjära ordinära differentialekvationer med tillämpningar inom interferensproblem i regulatoriska gennätverk

I detta examensarbete analyserar vi parameteroptimeringsproblem som är beskrivna med ordinära differentialekvationer (ODEer) och utvecklar beräkningstekniskt effektiva numeriska metoder för att beräkna lösningen. Dessutom härleder vi brusrobusta finita-differens approximationer för uppskattning av derivator i ODEn. De föreslagna metoderna har tillämpats för regulatoriska gennätverk (RGN).

RGNer är ansvariga för uttrycket av tusentals gener. Nätverksinferens handlar om att identifiera den komplicerad interaktionen mellan gener för att kunna karaktärisera cellernas tillstånd direkt från experimentella data. Tillförlitlig nätverksrekonstruktion är ett utmanande problem, trots att många metoder som använder många olika typer av konceptuella idéer har utvecklats. Detta beror på flera olika saker, inklusive att det finns ett enormt antal topologier, mycket brus, och komplexiteten av genregulering på olika nivåer. Ett lovande angreppssätt är dynamisk modellering från biologiska data som angriper en underliggande svaghet i den för tillfället ledande metoden för data-driven rekonstruktion. Metoden är beräkningstekniskt billig så att storleken på nätverket inte längre är huvudproblemet för beräkningen men ligger fortfarande i databegränsningar. Utmaningen är ett enormt antal av topologier. Därför bygger vi in ett filtreringssteg i metoder för att reducera antalet fria parameterar och simulerar sedan det dynamiska beteendet. Anledningen är att producera mer information om nätverkets struktur.

Vi utvärderar metoden på simulerat data, och studierar dess prestanda med avseende på datastorlek och brusnivå genom att tillämpa den på ett regulartoriskt gennätverk med 1565-gen E.coli. Vi illustrerar beräkningstiden över olika nätverksstorlekar och uppskattar beräkningskomplexiteten. Resultat på fem nätverk från DREAM4 är också presenterade och visar att vår metod har bättre prestanda än nuvarande metoder när de tillämpas på syntetiska data och tillåter rekonstruktion av bio-fysikaliskt noggranna dynamiska modeller från data med brus.

Contents

Chapter 1

Introduction

1.1 Background

1.1.1 Examples in general cases

Differential equations can appear in physical, chemical or biological models ranging from as simple as pendulum to as complex as Navier-Stokes equations in fluid dynamics. These differential equations often involve unknown parameters such as those shown in the *Exanples*. These parameters may have no physical meanings or are unlikely to be measured directly, so that the estimation of parameters in differential equations is crucial for simulation of the underlying physical, chemical or biological processes.

Examples

- Diffusion-reaction equation[1] with unknown diffusion coefficient D:

$$-D\Delta u + u = f$$

- FitzHugh-Nagumo model[2] characterizing neural spike potentials with parameters a, b, c unknown:

$$\dot{V} = c(V - \frac{V^3}{3} + R) + u(t)$$

$$\dot{R} = -\frac{1}{c}(V - a + bR)$$

Identification of parameters is often achieved by solving an optimization problem which minimizes the errors between the predictions of certain physical quantities

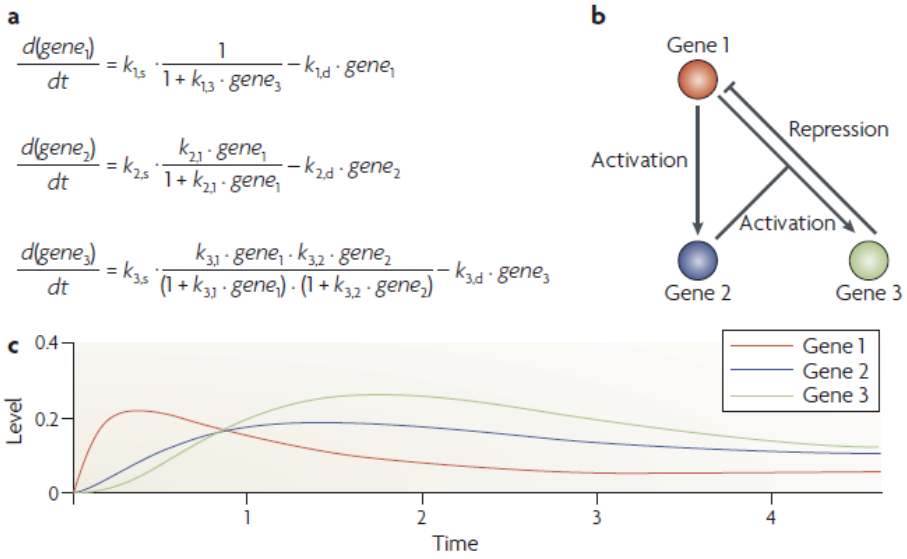


Figure 1.1: A gene network governed by ODEs (Figure Source: [3])

by differential equations and the observations of these quantities in experiments.

Many works have been done in this subject, J.O.Ramsay [2] applied the ideology of Finite Element Method (FEM) to identify the parameters in Ordinary Differential Equations (ODEs) and Vexler[1] analyzed an adaptive Finite Element Method to identify parameters in Partial Differential Equations (PDEs). These methods are widely applicable to all kinds of ODEs or PDEs, while they are so sophisticated that a long computation time would be taken if the amount of parameters are enormous.

Figure 1.1 (Source:[3]) illustrates a Gene Regulatory Network (GRN) with three genes modelled by an Ordinary Differential Equations (ODEs) system with 10 parameters. The size of GRNs can be remarkably large and thus the number of parameters to be identified increase quadratically. For instance, even in the simplest linear ODEs model, there are over 10,000 parameters for a 100-gene network. It took Kevin Y. Yip etc.[4] about 2 minutes, 13 hours, and 78 hours for prediction of the networks of size 10, 50 and 100, respectively. Therefore, the computation efficiency becomes a concern.

1.1.2 Gene Regulatory Network

A Gene Regulatory Network (GRN) is a network indicating the interactions between genes. The genes in a cell interact with each other by controlling expression level of RNA and proteins (Figure 1.2) and can be visualized as a directed graph

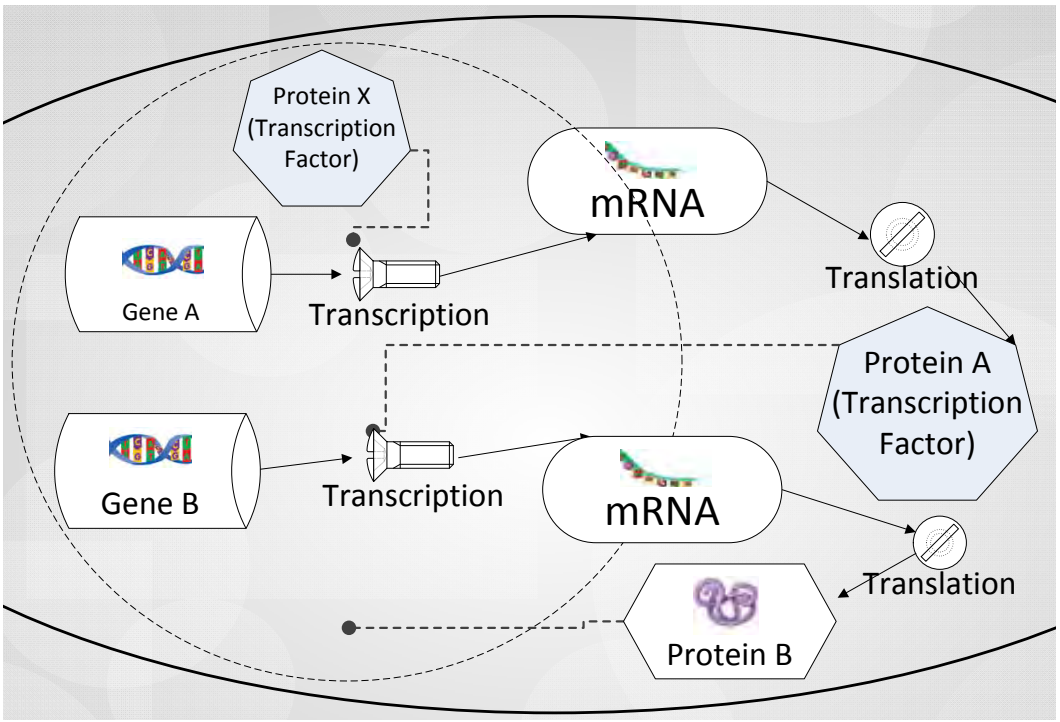


Figure 1.2: Interaction of genes in a cell. *Gene A* is transcribed to mRNA in nucleus and mRNA is translated into *Protein A* in cytoplasm. Certain proteins can induce or repress transcription of genes, which are called transcription factors. Gene B is regulated by the protein (transcription factor) controlled by Gene A

with genes as *nodes* and interactions as *arcs (directed edges)* as shown in Figure 1.3.

How genes regulate each other can be of great interest in biomedicine, bioinformatics and many other fields, and there have been many methods dealing with reconstruction of the gene regulatory network from experimental data. The mathematical models of gene regulation network (GRN) models range from logical models[5] with only Boolean values to continuous ones including detailed biochemical interactions[6]. Logical models require less biological details and computation complexity but also display limited dynamic behavior; on the contrast, concrete models describe more details of network dynamics while computational cost to determine parameters goes high.

Median-Corrected Z-Scores [7], Context Likelihood of Relatedness(CLR)[8] etc. can be applied to extract information of the network topology from steady-state data. The methods based on steady-state data face the inherent weaknesses that it is hard for them to distinguish between the direct interactions and indirect interactions, since the initial perturbation has spread into the network when the steady state is established. The linear ODE model[9], nonlinear ODE model[4] and non-

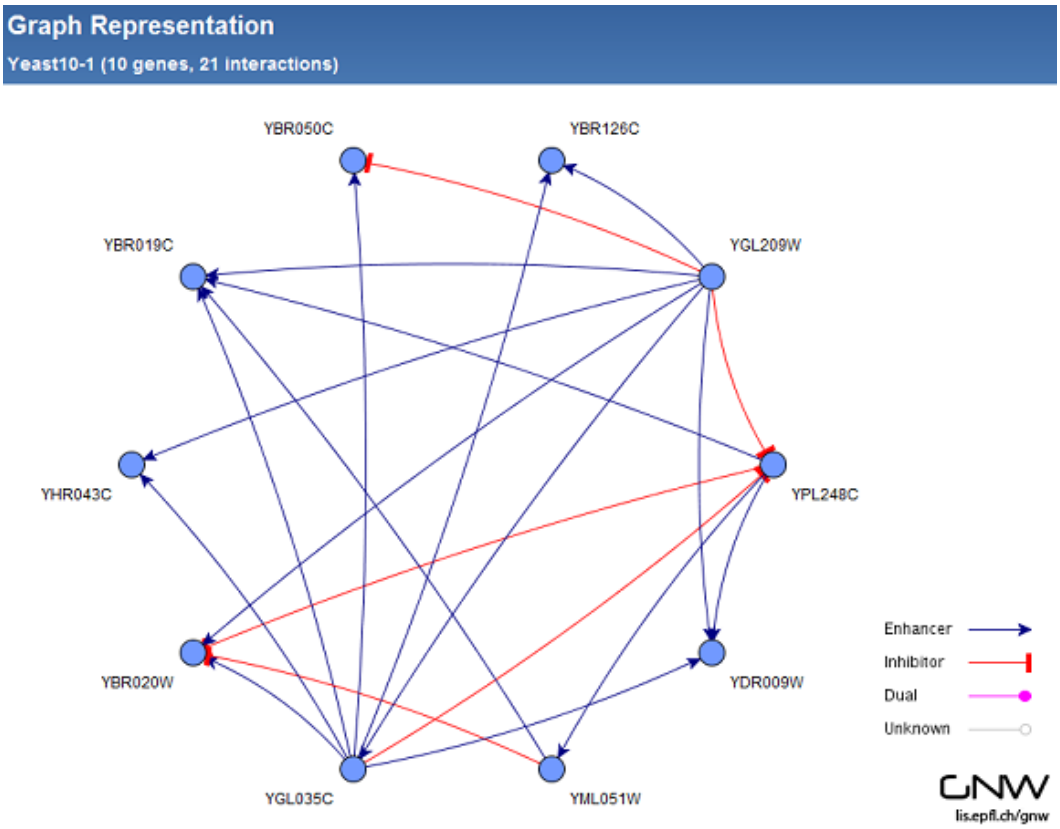


Figure 1.3: Network representation; produced by the software *GNW*. 10-node gene regulatory network extracted from a 4441-node GRN of Yeast.

parametric additive ODE model[10] have been developed to cope with time-series (dynamic) data. These methods have the ability to detect the transient perturbations in the network but with large amount parameters to be determined. There are also other methods based on machine learning[11], singular value decomposition (SVD)[12], Bayesian networks[13] and so forth.

Madar etc.[9] published a linear ODEs based method with filtration by CLR. Inspired by Madar, in this work, the linaer ODEs model is also applied and furthermore a computationally cheap algorithm will be proposed and a filtration based on a hypothesis test will be introduced. We choose the linear ODEs model describing the dynamics of the GRN and apply the proposed method in Chapter 2 to determine the parameters in the ODEs model, and thus reconstruct the topology of the network.

In this work, we present a method to identify extremely large amount of parameters in linear ODEs system. The change rates in expression level of a set of genes can

be described by a system of ODEs:

$$\frac{d\mathbf{x}}{dt} = \mathbf{a_0} + \mathbf{A}\mathbf{x}$$

where $\mathbf{x} \in \mathbb{R}^{n\times 1}, \mathbf{a_0} \in \mathbb{R}^{n\times 1}, \mathbf{A} \in \mathbb{R}^{n\times n}$ with $\mathbf{a_0}$ the basal expression rates, a_{ii} the self-decay rate and a_{ij} how the expression rate of gene-i is affected by other genes in the network.

For instance, the linear ODEs model of the network in Figure 1.1 can be written as

$$\begin{aligned}
\frac{dx_1}{dt} &= a_{01} + a_{11}x_1 + a_{12}x_2 + a_{13}x_3;\\
\frac{dx_2}{dt} &= a_{02} + a_{21}x_1 + a_{22}x_2 + a_{23}x_3;\\
\frac{dx_3}{dt} &= a_{03} + a_{31}x_1 + a_{32}x_2 + a_{33}x_3.
\end{aligned}$$

Since the ODEs are linear, it allows us to solve the optimization problem quite cheaply and thus enable us to determine large amount of parameters; moreover, the linear ODEs model is often not that bad for simulation of the real dynamics. It would be a good trade-off between the computation complexity and the model accuracy.

1.2 Organization of this thesis

The rest of this thesis is organized as follows.

Chapter 2 describes the optimization method. After a brief introduction of linear ODEs model and Frobenius norm in Section 2.1, two optimization problems for fitting the ODE parameters are discussed and solved in Section 2.2. A approach to estimate derivatives from noisy data is presented in Section 2.3.

In Chapter 3, we explain the application of the suggested method in reconstruction of the Gene Regulatory Networks (GRNs) and show the results of inferring a large *E.coli* gene regulatory network and estimate the computation time.

Chapter 4 provides a filtration method to reduce the model size of the ODEs model.

Chapter 5 shows the results on the DREAM project.

In Chapter 6, conclusions drawn from our methods are discussed.

Chapter 2

Identification of parameters in linear ODEs

2.1 Introduction

In this chapter, we describe the identification of the parameters in the linear Ordinary Differential Equations (linear ODEs) and give the theoretical solution by solving an optimization problem.

Generally speaking, we have a system of linear ODEs with unknown parameters and the goal is to find those parameters in a way that the modeled dynamics is consistent with given data. In section 2.1.1, we present the linear ODEs and the formulation of the problem. Section 2.1.2 gives the matrix notation of the given data. Section 2.2 is devoted to the theoretical solutions of both unconstrained and constrained optimization problems. In Section 2.3, we propose some numerical difference schemes for estimation of the derivatives in the ODEs.

2.1.1 Linear ODEs system

In general, a system of ODEs with parameters to be identified can be written as:

$$\frac{d\mathbf{x}}{dt} = f(t, \mathbf{x}; \boldsymbol{p}),$$

where $\mathbf{x} \in \mathbb{R}^n$ is a vector of state variables, n is the number of state variables; $f : [T \times \mathbb{R}^n] \to \mathbb{R}^n$, characterizes how the components in $\mathbf{x}$ interact with each other; $\boldsymbol{p} \in \mathbb{R}^{n_p}$ contains parameters to be fitted from experimental data.

In the linear case, this ODE system can be simply written as:

$$\frac{d\mathbf{x}}{dt} = \mathbf{a_0}^T + \mathbf{x}\mathbf{A}^T = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \begin{bmatrix} \mathbf{a_0}^T \\ \mathbf{A}^T \end{bmatrix}, \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^{1\times n}$, $\mathbf{a_0} \in \mathbb{R}^{n\times 1}$, $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{a_0}$ and $\mathbf{A}$ are parameters.

The problem is to find proper $\mathbf{a_0}$ and $\mathbf{A}$ such that the dynamic behavior of $\mathbf{x}(t)$ consists with observations in experiment.

Remark 1.

- Although $\mathbf{x}$ would be written in column vector $\tilde{\mathbf{x}} = \mathbf{x}^T \in \mathbb{R}^{n\times 1}$ in usual case as
$$\frac{d\tilde{\mathbf{x}}}{dt} = \mathbf{a_0} + \mathbf{A}\tilde{\mathbf{x}}, \tag{2.2}$$
the row vector $\mathbf{x} \in \mathbb{R}^{1\times n}$ is used in this thesis for convenience of notations in the following sections.

2.1.2 Matrices of time-series data

In order to identify the parameters, experimental data have to be provided. The time-series data of the observed subject can be obtained by beginning with perturbations from the steady state, and then a time course of changes $\mathbf{x}^{t_i} \in \mathbb{R}^n, i = 1, ..., T$ can be observed until the steady state has been rebuilt. The data of this single experiment can be recorded in a matrix:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{t_i} \\ \vdots \\ \mathbf{x}^{t_T} \end{bmatrix} \in \mathbb{R}^{T\times n}$$

Applying different perturbations, more experiments can be conducted in the same way and we call a series of repeated experiments conducted in the same way as *time course replicate experiments* or simply *replicates*, of which the r-th replicate can be denoted with a matrix form as:

$$\mathbf{X}_r \in \mathbb{R}^{T\times n}.$$

Therefore, all R replicates can be recorded in a series of matrices:

$$\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_r, ..., \mathbf{X}_R.$$

Furthermore, the observations of $\frac{d\mathbf{x}}{dt}$ in one experiment can be recorded following the same notation:

$$\mathbf{Y} = \begin{bmatrix} \frac{d\mathbf{x}}{dt}\Big|_{t_1} \\ \vdots \\ \frac{d\mathbf{x}}{dt}\Big|_{t_T} \end{bmatrix} \in \mathbb{R}^{T\times n};$$

as well as R replicates:

$$\mathbf{Y}_1, ..., \mathbf{Y}_r, ..., \mathbf{Y}_R.$$

These notations will enable us to form an optimization problem in the following section.

2.1.3 Frobenius Norm

We first introduce a matrix norm, the *Frobenius norm*, and its first order derivative and one property which will be employed later.

Definition 2.1.1 (Frobenius norm)**.** Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, then the *Frobenius norm* can be defined as:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}$$

Definition 2.1.2 (Frobenius norm)**.** Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, then the *Frobenius norm* can be also defined as:

$$\|\mathbf{A}\|_F = \sqrt{trace(\mathbf{A}^T\mathbf{A})}$$

One can easily show the two definitions are equivalent.

Lemma 2.1.1. *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, then the derivative of squared Frobenius norm of $\mathbf{A}$ with respect to $\mathbf{A}$ is:*

$$\frac{d\|\mathbf{A}\|_F^2}{d\mathbf{A}} = 2\mathbf{A}.$$

Proof. It can be easily proved by following the Definition 2.1.1.

$$\frac{\|\partial\mathbf{A}\|_F^2}{\partial a_{ij}} = \frac{\partial(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2)}{\partial a_{ij}} = \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{\partial(a_{ij}^2)}{\partial a_{ij}} = 2a_{ij}.$$

or in matrix form:

$$\frac{d\|\mathbf{A}\|_F^2}{d\mathbf{A}} = 2\mathbf{A}.$$

□

Lemma 2.1.2. *Let there be some s matrices with the same column size $A_1 \in \mathbb{R}^{m_1 \times n}, ..., A_s \in \mathbb{R}^{m_s \times n}$ and $\mathbf{B} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_s \end{bmatrix}$, then*

$$\sum_{i=1}^{s}\|\mathbf{A}_i\|_F^2 = \|\mathbf{B}\|_F^2.$$

Proof. From the Definition 2.1.2, we have

$$\begin{aligned}\|\mathbf{B}\|_F^2 &= trace(\mathbf{B}^T\mathbf{B}) \\ &= trace(\begin{bmatrix}\mathbf{A}_1^T \dots \mathbf{A}_s^T\end{bmatrix}\begin{bmatrix}\mathbf{A}_1 \\ \vdots \\ \mathbf{A}_s\end{bmatrix}) \\ &= trace(\sum_{i=1}^{s}\mathbf{A}_i^T\mathbf{A}_i) \\ &= \sum_{i=1}^{s} trace(\mathbf{A}_i^T\mathbf{A}_i) = \sum_{i=1}^{s}\|\mathbf{A}_i\|_F^2.\end{aligned}$$

□

2.2 Parameter optimization

2.2.1 Unconstrained optimization in Frobenius norm

With the properties in Section 2.1.3 , we can deduce the parameter identification problem into a minimization problem in Frobenius norm.

The distance between the experimental observations of $d\mathbf{x}/dt$ recorded in

$$\mathbf{Y}_r \in \mathbb{R}^{T\times n}$$

and the hypothesis in the linear ODEs system (Equation (2.1))

$$h(\tilde{\mathbf{A}}; \mathbf{X}_r) = \mathbf{a_0}^T + \mathbf{X}_r\mathbf{A}^T = [\mathbf{1}\ \mathbf{X}_r]\begin{bmatrix}\mathbf{a_0}^T \\ \mathbf{A}^T\end{bmatrix}$$

can be written in the sense of Frobenius norm

$$\epsilon_r^2 = \|\mathbf{Y}_r - h(\tilde{\mathbf{A}}; \mathbf{X}_r)\|_F^2 \tag{2.3}$$

where $\tilde{\mathbf{A}} = [\mathbf{a_0}\ \mathbf{A}] \in \mathbb{R}^{n\times(n+1)}$ and $\mathbf{1} \in \mathbb{R}^{n\times 1}$ with all elements are ones.

Thus, the objective function to be minimized can be written as a summation of ϵ_r^2 over all replicate experiments:

$$J(\tilde{\mathbf{A}}) = \frac{1}{2R}\sum_{i=1}^{R}\|\mathbf{Y}_r - h(\tilde{\mathbf{A}}; \mathbf{X}_r)\|_F^2. \tag{2.4}$$

where R is the number of replicate experiments.

Theorem 2.2.1. *The objective function in equation (2.4) can be written in a single Frobenius norm as:*

$$J(\tilde{\mathbf{A}}) = \frac{1}{2R}|\mathbf{D_y} - \mathbf{D_x}\tilde{\mathbf{A}}^T\|_F^2. \tag{2.5}$$

where

$$\mathbf{D_y} = \begin{bmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_r \\ \vdots \\ \mathbf{Y}_R \end{bmatrix} \quad \textit{and} \quad \mathbf{D_x} = \begin{bmatrix} \mathbf{1} & \mathbf{X}_1 \\ \vdots & \vdots \\ \mathbf{1} & \mathbf{X}_r \\ \vdots & \vdots \\ \mathbf{1} & \mathbf{X}_R \end{bmatrix}.$$

Proof. From the Lemma 2.1.2, we have :

$$\begin{aligned} J(\tilde{\mathbf{A}}) &= \frac{1}{2R}\sum_{i=1}^{R}\|\mathbf{Y}_r - h(\tilde{\mathbf{A}};\mathbf{X}_r)\|_F^2 \\ &= \frac{1}{2R}\left\|\begin{bmatrix} \mathbf{Y}_1 - h(\tilde{\mathbf{A}};\mathbf{X}_1) \\ \vdots \\ \mathbf{Y}_R - h(\tilde{\mathbf{A}};\mathbf{X}_R) \end{bmatrix}\right\|_F^2 \\ &= \frac{1}{2R}\left\|\begin{bmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_R \end{bmatrix} - \begin{bmatrix} h(\tilde{\mathbf{A}};\mathbf{X}_1) \\ \vdots \\ h(\tilde{\mathbf{A}};\mathbf{X}_R) \end{bmatrix}\right\|_F^2 \\ &= \frac{1}{2R}\left\|\begin{bmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_R \end{bmatrix} - \begin{bmatrix} \mathbf{1} & \mathbf{X}_1 \\ \vdots & \vdots \\ \mathbf{1} & \mathbf{X}_R \end{bmatrix}\tilde{\mathbf{A}}^T\right\|_F^2 \\ &= \frac{1}{2R}\|\mathbf{D_y} - \mathbf{D_x}\tilde{\mathbf{A}}^T\|_F^2. \end{aligned}$$

□

Therefore, the minimization problem can be written as:

$$\begin{aligned} &\text{Find } \tilde{\mathbf{A}} \in \mathbb{R}^{n\times(n+1)}, \text{ such that} \\ &\qquad J(\tilde{\mathbf{A}}) = \frac{1}{2R}\|\mathbf{D_y} - \mathbf{D_x}\tilde{\mathbf{A}}^T\|_F^2 \ \text{ is minimized.} \end{aligned} \tag{2.6}$$

Remark 2.

- Since Y_r records the data of $d\mathbf{x}/dt$, we call $\mathbf{D_y} = \begin{bmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_R \end{bmatrix} \in \mathbb{R}^{R\cdot T\times n}$ the Derivative matrix and $\mathbf{D_x} = \begin{bmatrix} \mathbf{1} & \mathbf{X}_1 \\ \vdots & \vdots \\ \mathbf{1} & \mathbf{X}_R \end{bmatrix} \in \mathbb{R}^{R\cdot T\times(n+1)}$ the Design matrix,

which can be re-designed into higher order such as $\mathbf{D_x} = \begin{bmatrix} \mathbf{1} & \mathbf{X}_1 & \mathbf{X}_1^2 & \dots \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{1} & \mathbf{X}_R & \mathbf{X}_R^2 & \dots \end{bmatrix}$ without altering the linearity of the objective function $J(\tilde{\mathbf{A}})$.

- The hypothesis $h(\tilde{\mathbf{A}}; \mathbf{X}_r)$ is always linear with respect to $\tilde{\mathbf{A}}$ so that $J(\tilde{\mathbf{A}})$ is quadratic and convex; hence the global minimum can be easily found by solving a normal equation, usually via QR factorization.
- A regularization term can be applied:

$$J(\tilde{\mathbf{A}}) = \frac{1}{2R}\|\mathbf{D_y} - \mathbf{D_x}\tilde{\mathbf{A}}^T\|_F^2 + \frac{\alpha}{2R}\|\mathbf{A}\|_F^2; \tag{2.7}$$

  in which α can be determined via cross validation.

It has been well known that a zero gradient gives the solution of the problem in equation (2.6):

$$\frac{dJ(\tilde{\mathbf{A}})}{d\tilde{\mathbf{A}}^T} = \mathbf{0}$$

Theorem 2.2.2. *The solution of*

$$\arg\min_{\tilde{\mathbf{A}}} J(\tilde{\mathbf{A}}) = \frac{1}{2R}\|\mathbf{D_y} - \mathbf{D_x}\tilde{\mathbf{A}}^T\|_F^2 + \frac{\alpha}{2R}\|\mathbf{A}\|_F^2$$

is

$$\tilde{\mathbf{A}} = \mathbf{D_y}^T\mathbf{D_x}(\mathbf{D_x}^T\mathbf{D_x} + \alpha\hat{\mathbf{E}})^{-T}.$$

where

$$\hat{\mathbf{E}} = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}.$$

Proof. Firstly, by applying Lemma 2.1.1 and chain rule to the first term of $J(\tilde{\mathbf{A}})$, we have:

$$\begin{aligned} \frac{1}{2R}\frac{d}{d\tilde{\mathbf{A}}^T}\|\mathbf{D_y} - \mathbf{D_x}\tilde{\mathbf{A}}^T\|_F^2 &= \frac{1}{2R}\left(\frac{d(-\mathbf{D_x})\tilde{\mathbf{A}}^T)}{d\tilde{\mathbf{A}}^T}\right)^T 2(\mathbf{D_y} - \mathbf{D_x}\tilde{\mathbf{A}}^T) \\ &= \frac{1}{2R}(-\mathbf{D_x})^T 2(\mathbf{D_y} - \mathbf{D_x}\tilde{\mathbf{A}}^T) \\ &= \frac{1}{R}\mathbf{D_x}^T\mathbf{D_x}\tilde{\mathbf{A}}^T - \frac{1}{R}\mathbf{D_x}^T\mathbf{D_y} \end{aligned}$$

Then, we re-write the second term on the right hand side into:

$$\|\mathbf{A}\|_F^2 = \|\mathbf{A}^T\|_F^2 = \left\|\begin{bmatrix} \mathbf{0} \\ \mathbf{A}^T \end{bmatrix}\right\|_F^2 := \|\hat{\mathbf{A}}^T\|_F^2$$

and as mentioned in Equation 2.3:

$$\tilde{\mathbf{A}}^T = \begin{bmatrix} \mathbf{a_0}^T \\ \mathbf{A}^T \end{bmatrix}$$

So, we have

$$\left[\frac{d\|\mathbf{A}\|_F^2}{d\tilde{\mathbf{A}}^T}\right]_{ij} = \frac{d\|\mathbf{A}\|_F^2}{d\tilde{a}_{ji}} = \frac{d\|\hat{\mathbf{A}}^T\|_F^2}{d\tilde{a}_{ji}} = \begin{cases} 0 & \text{if } i = 1 \\ \tilde{a}_{ji} & \text{if } i \neq 1 \end{cases}$$

or in matrix form

$$\frac{d\|\mathbf{A}\|_F^2}{d\tilde{a}_{ij}} = 2\hat{\mathbf{E}}\tilde{\mathbf{A}}^T$$

where

$$\hat{\mathbf{E}} = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}.$$

Combine the two terms above, we have:

$$\mathbf{0} = \frac{dJ(\tilde{\mathbf{A}})}{d\tilde{\mathbf{A}}^T} = \frac{1}{R}\mathbf{D_x}^T\mathbf{D_x}\tilde{\mathbf{A}}^T - \frac{1}{R}\mathbf{D_x}^T\mathbf{D_y} + \frac{\alpha}{R}\hat{\mathbf{E}}\tilde{\mathbf{A}}^T$$

or

$$(\mathbf{D_x}^T\mathbf{D_x} + \alpha\hat{\mathbf{E}})\tilde{\mathbf{A}}^T = \mathbf{D_x}^T\mathbf{D_y} \tag{2.8}$$

Therefore, the solution can be written as:

$$\tilde{\mathbf{A}} = \mathbf{D_y}^T\mathbf{D_x}(\mathbf{D_x}^T\mathbf{D_x} + \alpha\hat{\mathbf{E}})^{-T}.$$

□

Remark 3.

- The solution above is theoretically accurate; however, $\mathbf{D_x^T D_x}$ is usually ill-conditioned, since its condition number is amplified to

  $$\kappa(\mathbf{D_x^T D_x}) = \kappa(\mathbf{D_x})^2$$

  which may lead to an unacceptably large error when numerical methods are applied.

- The approach of QR factorization is more stable and recommended[14]. The normal equation 2.8 can be rewritten into the following form:

  $$\begin{bmatrix} \mathbf{D_x} \\ \sqrt{\alpha}\hat{\mathbf{E}} \end{bmatrix}^T \begin{bmatrix} \mathbf{D_x} \\ \sqrt{\alpha}\hat{\mathbf{E}} \end{bmatrix} \tilde{\mathbf{A}}^T = \begin{bmatrix} \mathbf{D_x} \\ \sqrt{\alpha}\hat{\mathbf{E}} \end{bmatrix}^T \begin{bmatrix} \mathbf{D_y} \\ \mathbf{0} \end{bmatrix}$$

  and the approach of QR factorization can be applied to solve:

  $$\begin{bmatrix} \mathbf{D_x} \\ \sqrt{\alpha}\hat{\mathbf{E}} \end{bmatrix} \tilde{\mathbf{A}}^T = \begin{bmatrix} \mathbf{D_y} \\ \mathbf{0} \end{bmatrix}.$$

2.2.2 Constrained optimization in Frobenius norm

Sometimes people have already obtained prior knowledge about the ODE system that some parameters are zero; or before the fitting of ODEs, other methods have been applied and some unlikely nonzero parameters have been filtered out; we will discuss one method to do the filtration in Section 4.2.1.

In these situations, certain parameters in the ODE model have to be restricted to zero and mathematically it becomes an *equality constrained optimization problem*:

$$\begin{aligned} \min_{\tilde{\mathbf{A}}\in\mathbb{R}^{n\times(n+1)}} J(\tilde{\mathbf{A}}) := \frac{1}{2R}\|\mathbf{D_y} - \mathbf{D_x}\tilde{\mathbf{A}}^T\|_F^2 + \frac{\alpha}{2R}\|\mathbf{A}\|_F^2 \ \text{ subject to} \\ a_{kl} = 0,\ \forall (k,l) \in \boldsymbol{C} \end{aligned} \tag{2.9}$$

where a_{kl} is an element in $\mathbf{A}$ and $\boldsymbol{C} \subset \{(i,j)|i,j \in \{1,..,n\}\}$ contains all zero constraints.

The most popular approach to solve equality constrained optimization problem is the method of *Lagrange multipliers* (λ). We introduce this new variable λ into the objective function 2.9 which is then called a *Lagrange function* (or *Lagrangian*):

$$L(\tilde{\mathbf{A}}, \lambda) = \frac{1}{2R}\|\mathbf{D_y} - \mathbf{D_x}\tilde{\mathbf{A}}^T\|_F^2 + \frac{\alpha}{2R}\|\mathbf{A}\|_F^2 + \frac{1}{R}\sum_{(k,l)\in\boldsymbol{C}} \lambda_{kl} a_{kl} \tag{2.10}$$

and the solution of the linear system gives out the globle minimum point

$$\begin{cases} \frac{\partial L(\tilde{\mathbf{A}},\lambda)}{\partial \tilde{\mathbf{A}}^T} &= \mathbf{0} \\ \frac{\partial L(\tilde{\mathbf{A}},\lambda)}{\partial \lambda_{kl}} &= 0,\ \forall (k,l) \in \boldsymbol{C} \end{cases}. \tag{2.11}$$

In order to solve this linear system, we first introduce a *vectorization operator* and one of its properties used later.

Definition 2.2.1 (vectorization operator)**.** Let $\mathbf{A} = [\mathbf{a_1}, ..., \mathbf{a_i}, ..., \mathbf{a_n}] \in \mathbb{R}^{m\times n}$ and $\mathbf{a_i} \in \mathbb{R}^{m\times 1}$ be the i-th column of $\mathbf{A}$, the *vectorization operator* $vec : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn\times 1}$ maps A into a column vector by queuing the column vectors of $\mathbf{A}$ to the rear of the queue one by another:

$$vec(\mathbf{A}) = \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \\ \vdots \\ \mathbf{a_n} \end{bmatrix} \in \mathbb{R}^{mn\times 1}.$$

Lemma 2.2.3. *Let* $\mathbf{A} \in \mathbb{R}^{m\times l}, \mathbf{B} \in \mathbb{R}^{l\times n}$*, then*

$$vec(\mathbf{AB}) = (\mathbf{I}_n \otimes \mathbf{A})vec(\mathbf{B}),$$

where $\otimes$ *is Kronecker product or tensor product.*

Proof. Let B be partitioned by columns

$$\mathbf{B} = [\mathbf{b_1}, ...\mathbf{b_i}, ..., \mathbf{b_n}],\ \mathbf{b_i} \in \mathbb{R}^{l\times 1}.$$

We have

$$\mathbf{AB} = [\mathbf{Ab_1}, ..., \mathbf{Ab_n}]$$

From the Definition 2.2.1,

$$vec(\mathbf{AB}) = \begin{bmatrix} \mathbf{Ab_1} \\ \mathbf{Ab_2} \\ \vdots \\ \mathbf{Ab_n} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & & & \\ & \mathbf{A} & & \\ & & \ddots & \\ & & & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{b_1} \\ \mathbf{b_2} \\ \vdots \\ \mathbf{b_n} \end{bmatrix} = (\mathbf{I}_n \otimes \mathbf{A})vec(\mathbf{B}).$$

□

With the vectorization operator, the linear system (2.2.2) can be written into a matrix form and solved at once.

Theorem 2.2.4. *To solve the linear system (2.2.2) is equivalent to solve the following linear system:*

$$\begin{bmatrix} \mathbf{P} & \mathbf{E_C} \\ \mathbf{E_C}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} vec(\tilde{\mathbf{A}}^T) \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} vec(\mathbf{D_x}^T\mathbf{D_y}) \\ \mathbf{0} \end{bmatrix}$$

where

$$\begin{aligned} \mathbf{P} &= \mathbf{I}_{n+1} \otimes \left(\mathbf{D_x}^T\mathbf{D_x} + \alpha\hat{\mathbf{E}}\right) \in \mathbb{R}^{(n+1)^2\times(n+1)^2}, \\ \mathbf{E_C} &= [..., vec(\mathbf{E}_{kl}), ...] \in \mathbb{R}^{(n+1)^2\times|\boldsymbol{C}|}, \\ \boldsymbol{\lambda} &= [..., \lambda_{kl}, ...]^T \in \mathbb{R}^{|\boldsymbol{C}|\times 1}, \\ \mathbf{E}_{kl} &= [e_{ij}] \in \mathbb{R}^{(n+1)\times n} \text{ with } e_{ij} = \delta_i^k\delta_j^l, i = 0, 1, .., n, j = 1, ..., n. \end{aligned}$$

in which $(k, l) \in \boldsymbol{C}$, $|\boldsymbol{C}|$ is the number of elements or cardinality of set $\boldsymbol{C}$ and δ_i^j is the Kronecker delta.

Proof. To solve the linear system (2.2.2):

$$\begin{cases} \frac{\partial L(\tilde{\mathbf{A}},\lambda)}{\partial \tilde{\mathbf{A}}^T} &= \mathbf{0} \\ \frac{\partial L(\tilde{\mathbf{A}},\lambda)}{\partial \lambda_{kl}} &= 0,\ \forall(k, l) \in \boldsymbol{C} \end{cases}$$

we first have to calculate the partial direvative of the Lagrange function (2.10):

$$L(\tilde{\mathbf{A}}, \lambda) = \frac{1}{2R}\|\mathbf{D_y} - \mathbf{D_x}\tilde{\mathbf{A}}^T\|_F^2 + \frac{\alpha}{2R}\|\mathbf{A}\|_F^2 + \frac{1}{R}\sum_{(k,l)\in\boldsymbol{C}} \lambda_{kl}a_{kl}$$

For the first two terms of $\frac{\partial L(\tilde{\mathbf{A}},\lambda)}{\partial \tilde{\mathbf{A}}^T}$, we have already known from the proof of Theorem 2.2.2

$$\frac{d}{d\tilde{\mathbf{A}}^T}\left(\frac{1}{2R}\|\mathbf{D_y}-\mathbf{D_x}\tilde{\mathbf{A}}^T\|_F^2+\frac{\alpha}{2R}\|\mathbf{A}\|_F^2\right)$$

$$=\frac{1}{R}\mathbf{D_x}^T\mathbf{D_x}\tilde{\mathbf{A}}^T-\frac{1}{R}\mathbf{D_x}^T\mathbf{D_y}+\frac{\alpha}{R}\hat{\mathbf{E}}\tilde{\mathbf{A}}^T$$

where

$$\hat{\mathbf{E}}=\begin{bmatrix}0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1\end{bmatrix}\in\mathbb{R}^{(n+1)\times(n+1)}.$$

Differentiating the third term, we have

$$\begin{aligned}&\left[\frac{d}{d\tilde{\mathbf{A}}^T}\left(\frac{1}{R}\sum_{(k,l)\in\boldsymbol{C}}\lambda_{kl}a_{kl}\right)\right]_{ij}\\ &=\frac{1}{R}\sum_{(k,l)\in\boldsymbol{C}}\left(\lambda_{kl}\frac{da_{kl}}{da_{ij}}\right)\\ &=\frac{1}{R}\sum_{(k,l)\in\boldsymbol{C}}\left(\lambda_{kl}\delta_i^k\delta_j^l\right),\ i=0,1,..,n,j=1,...,n\end{aligned}$$

or in matrix form:

$$\frac{d}{d\tilde{\mathbf{A}}^T}\left(\frac{1}{R}\sum_{(k,l)\in\boldsymbol{C}}\lambda_{kl}a_{kl}\right)=\frac{1}{R}\sum_{(k,l)\in\boldsymbol{C}}\lambda_{kl}\mathbf{E}_{kl}$$

where

$$\mathbf{E}_{kl}=[e_{ij}]\in\mathbb{R}^{(n+1)\times n}\text{ with }e_{ij}=\delta_i^k\delta_j^l, i=0,1,..,n,j=1,...,n.$$

Therefore, we have:

$$\mathbf{0}=\frac{\partial L(\tilde{\mathbf{A}},\lambda)}{\partial\tilde{\mathbf{A}}^T}=\frac{1}{R}\mathbf{D_x}^T\mathbf{D_x}\tilde{\mathbf{A}}^T-\frac{1}{R}\mathbf{D_x}^T\mathbf{D_y}+\frac{\alpha}{R}\hat{\mathbf{E}}\tilde{\mathbf{A}}^T+\frac{1}{R}\sum_{(k,l)\in\boldsymbol{C}}\lambda_{kl}\mathbf{E}_{kl}$$

or

$$\left(\mathbf{D_x}^T\mathbf{D_x}+\alpha\hat{\mathbf{E}}\right)\tilde{\mathbf{A}}^T+\sum_{(k,l)\in\boldsymbol{C}}\lambda_{kl}\mathbf{E}_{kl}=\mathbf{D_x}^T\mathbf{D_y}\tag{2.12}$$

By applying the vectorization operator to equation (2.12), and from Lemma 2.2.3, we have:

$$\mathbf{I}_{n+1}\otimes\left(\mathbf{D_x}^T\mathbf{D_x}+\alpha\hat{\mathbf{E}}\right)vec(\tilde{\mathbf{A}}^T)+\sum_{(k,l)\in\boldsymbol{C}}\lambda_{kl}vec(\mathbf{E}_{kl})=vec(\mathbf{D_x}^T\mathbf{D_y})\tag{2.13}$$

Note that $vec(\mathbf{E}_{kl})$ is a column vector and $\sum_{(k,l)\in \boldsymbol{C}} \lambda_{kl} vec(\mathbf{E}_{kl})$ is a linear combination of $vec(\mathbf{E}_{kl})$, then it can be written into a matrix form:

$$[..., vec(\mathbf{E}_{kl}), ...] \begin{bmatrix} \vdots \\ \lambda_{kl} \\ \vdots \end{bmatrix} := \mathbf{E}_{\boldsymbol{C}} \boldsymbol{\lambda}, \text{ with } (k,l) \in \boldsymbol{C}$$

Therefore, the equation (2.13) can be written as:

$$\left(\mathbf{D_x}^T \mathbf{D_x} + \alpha \hat{\mathbf{E}}\right) \tilde{\mathbf{A}}^T + \mathbf{E}_{\boldsymbol{C}} \boldsymbol{\lambda} = \mathbf{D_x}^T \mathbf{D_y} \tag{2.14}$$

For the second equation in equation (2.2.2): $\frac{\partial L(\tilde{\mathbf{A}}, \lambda)}{\partial \lambda_{kl}} = 0, \ \forall (k,l) \in \boldsymbol{C}$,we have:

$$\frac{\partial L(\tilde{\mathbf{A}}, \lambda)}{\partial \lambda_{kl}} = \frac{1}{R} \sum_{(k,l)\in \boldsymbol{C}} \frac{d}{d\lambda_{kl}} \lambda_{kl} a_{kl} = \frac{1}{R} a_{kl} = 0, \ \forall (k,l) \in \boldsymbol{C}$$

Note that

$$vec(\mathbf{E}_{kl})^T vec(\tilde{\mathbf{A}}^T) = a_{kl}$$

Then, we have:

$$\frac{\partial L(\tilde{\mathbf{A}}, \lambda)}{\partial \lambda_{kl}} = vec(\mathbf{E}_{kl})^T vec(\tilde{\mathbf{A}}^T) = 0, \ \forall (k,l) \in \boldsymbol{C}$$

or in matrix form:

$$\begin{bmatrix} \vdots \\ vec(\mathbf{E}_{kl})^T \\ \vdots \end{bmatrix} vec(\tilde{\mathbf{A}}^T) = \begin{bmatrix} \vdots \\ 0 \\ \vdots \end{bmatrix} = \mathbf{E}_{\boldsymbol{C}}^T vec(\tilde{\mathbf{A}}^T) \tag{2.15}$$

Assembling equation (2.14) and equation (2.15) into a martix form, we have:

$$\begin{bmatrix} \mathbf{I}_{n+1} \otimes \left(\mathbf{D_x}^T \mathbf{D_x} + \alpha \hat{\mathbf{E}}\right) & \mathbf{E}_{\boldsymbol{C}} \\ \mathbf{E}_{\boldsymbol{C}}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} vec(\tilde{\mathbf{A}}^T) \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} vec(\mathbf{D_x}^T \mathbf{D_y}) \\ \mathbf{0} \end{bmatrix}.$$

□

Remark 4.

- The coefficient matrix
$$\begin{bmatrix} \mathbf{P} & \mathbf{E}_{\boldsymbol{C}} \\ \mathbf{E}_{\boldsymbol{C}}^T & \mathbf{0} \end{bmatrix}$$
is called *Karush-Kuhn-Tucker (KKT)* matrix, and it is nonsingular if and only if $\mathbf{P} + \mathbf{E_C E_C^T}$ is positive definite.

2.3 Numerical Differentiation of Noisy Data

In the above sections, we stated that the derivative $d\mathbf{x}/dt$ could be measured directly from the experiments, for instance the Doppler radar extracts the velocities (the derivative of the position) of the targets. However, it is not always the case, and then the derivatives need to be estimated from the observed $\mathbf{x}$ which can be noisy due to measurement errors.

There have been many works dealing with numerical differentiation of noisy data[15][16][17], and here we focus on finite difference formulas. We first take the *explicit Euler scheme* and *3-point central difference scheme* as examples and show why the former is not a good choice; thereafter, a series of better schemes will be proposed.

2.3.1 Two examples

In many articles[12][9] , the explicit Euler's scheme

$$\frac{d\mathbf{x}}{dt} = \frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h} + O(h)$$

was applied which though easy to implement, has limitation on step size h due to numerical stability concerns[18] and drawback of amplifying noise level.

Example 2.3.1 (noise amplification by Euler's scheme)**.**
Let the measurement errors $\boldsymbol{\varepsilon}$ of $\mathbf{x}$ be independent and identically distributed ($i.i.d$) Gaussian noises with mean $\mu = 0$ and unknown variance σ^2:

$$\begin{aligned} \mathbf{x} &= \bar{\mathbf{x}} + \boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon} &\sim i.i.d.\ \mathcal{N}(0, \sigma^2) \end{aligned}$$

then, we have

$$\frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h} = \frac{\bar{\mathbf{x}}(t+h) - \bar{\mathbf{x}}(t)}{h} + \frac{\boldsymbol{\varepsilon}_1 - \boldsymbol{\varepsilon}_0}{h}.$$

Since $\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_0$ are $i.i.d\ \mathcal{N}(0, \sigma^2)$, the variance of the noise of estimated differentiation becomes:

$$\sigma_D^2 = Var\left(\frac{\boldsymbol{\varepsilon}_1 - \boldsymbol{\varepsilon}_0}{h}\right) = \frac{Var(\boldsymbol{\varepsilon}_1)}{h^2} + \frac{Var(\boldsymbol{\varepsilon}_0)}{h^2} = 2\frac{\sigma^2}{h^2}.$$

For comparison, we take one more well-known scheme as another example.

Example 2.3.2 (noise amplification by 3-point central difference scheme)**.**
3-point central difference scheme:

$$\frac{d\mathbf{x}}{dt} = \frac{\mathbf{x}(t+h) - \mathbf{x}(t-h)}{2h} + O(h^2),$$

with the variance of the noise of estimated differentiation

$$\sigma_D^2 = Var\left(\frac{\boldsymbol{\varepsilon}_1 - \boldsymbol{\varepsilon}_{-1}}{2h}\right) = \frac{Var(\boldsymbol{\varepsilon}_1)}{4h^2} + \frac{Var(\boldsymbol{\varepsilon}_{-1})}{4h^2} = \frac{1}{2}\frac{\sigma^2}{h^2}.$$

These two examples show that the Euler's formula will amplify the noise level as four times as that of 3-point central difference formula, given the same noisy data.

2.3.2 More Central-Difference Formulas

Finite difference schemes can be deduced from Taylor expansion, polynomial interpolation or polynomial fitting etc.. Since we are dealing with noisy data, in this subsection we will concentrate on polynomial fitting rather than interpolation. Moreover, we only discuss central difference schemes which yield higher accuracy[19].

The main idea is to fit a polynomial locally with a few neighbor points and then differentiate the fitted polynomial theoretically.

Theorem 2.3.1 (Fitted Central Derivative Scheme)**.** *Let $P_n(t)$ be a polynomial of order n:*

$$P_n(t) = a_0 + a_1(t - t_0) + ... + a_n(t - t_0)^n,$$

fitted into $(t_0, x(t_0))$ and its $m = 2k$ neighbor nodes:

$$\begin{array}{cccccccc} t_0 - kh & ... & t_0 - h & t_0 & t_0 + h & ... & t_0 + kh \\ x(t_0 - kh) & ... & x(t_0 - h) & x(t_0) & x(t_0 + h) & ... & x(t_0 + kh) \end{array}$$

then the derivative of x at the point $t = t_0$ can be approximated by a_1, which can be solved from the following linear system:

$$(V^T V + \lambda I) \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = V^T \begin{bmatrix} x(t_0 - kh) \\ \vdots \\ x(t_0) \\ \vdots \\ x(t_0 + kh) \end{bmatrix} \tag{2.16}$$

where V is the Vandermonde matrix

$$V = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & -ih & (-ih)^2 & \dots & (-ih)^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \in \mathbb{R}^{(m+1)\times(n+1)},\ i = -k, ..., 0, ..., k$$

Proof. The parameters in the polynomial can be fitted by the classical linear least squares regression:

$$\min_{a_0, a_1, ..., a_n \in \mathbb{R}} \frac{1}{2} \sum_{i=-k}^{k} |x(t - ik) - P_n(t - ih)|^2 + \frac{\lambda}{2} \sum_{i=0}^{n} a_i^2$$

of which the solution has been well known as shown in equation (2.16) in the this theorem. Furthermore, the estimation of dx/dt is:

$$\left.\frac{d\mathbf{x}}{dt}\right|_{t=t_0} \approx \left.\frac{P_n(t)}{dt}\right|_{t=t_0} = a_1$$

□

Remark 5.

- The regularization term $\lambda > 0$ is usually called smoothing parameter, with larger λ indicating a smoother fitted curve but less fidelity to the data.
- If $m \geq n$, the solution is unique. If $m = n$ and $\lambda = 0$, then the polynomial fitting collapses to Lagrange interpolation, which yields the calssical $(m+1)$-point central difference scheme. As the fitted curve goes exactly through data points in interpolation, it will not be a good choice when data is noisy.
- m is the window length of the moving fitting, indicating how many nodes are involved; n is the order of fitted polynomial and shows the accuracy of the difference scheme. Once m and n are given, the scheme can be determined, which can be called *Fitted Central Derivative Scheme*, denoted as $FCDS(m, n)$

We calculate $FCDS(2, 2)$ as an example.

Example 2.3.3 (FCDS(2,2))**.** Since $m = 2k = 2,\ n = 2$, follwing the Theorem 2.3.1, one can write down the Vandermonde matrix:

$$V = \begin{bmatrix} 1 & -h & h^2 \\ 1 & 0 & 0 \\ 1 & h & h^2 \end{bmatrix}$$

and the normal equation:

$$(V^T V + \lambda I) \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = V^T \begin{bmatrix} x(t_0 - h) \\ x(t_0) \\ x(t_0 + h) \end{bmatrix}$$

After solving this equation, we have:

$$\left.\frac{d\mathbf{x}}{dt}\right|_{t=t_0} \approx \left.\frac{P_n(t)}{dt}\right|_{t=t_0} = a_1 = \frac{\mathbf{x}(t_0 + h) - \mathbf{x}(t_0 - h)}{2h + \lambda/h}.$$

If $\lambda = 0$, since $m = n = 2$, it becomes a Lagrange interpolation and the scheme is the classical 3-point central-difference formula as shown in Example 2.3.2:

$$\left.\frac{d\mathbf{x}}{dt}\right|_{t=t_0} \approx \frac{\mathbf{x}(t_0 + h) - \mathbf{x}(t_0 - h)}{2h}.$$

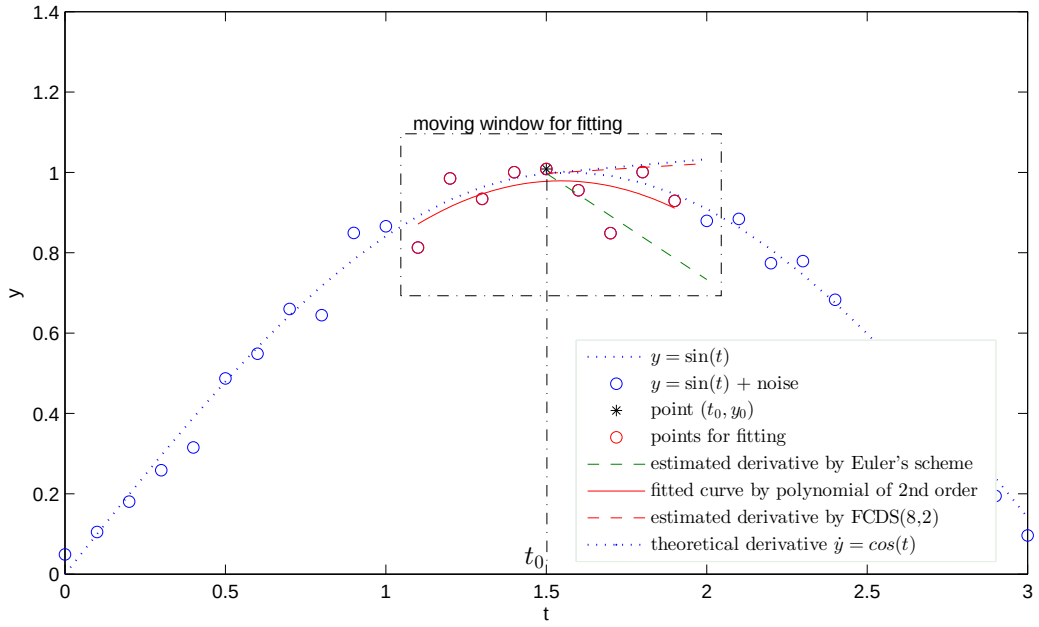


Figure 2.1: An example of derivative estimated by polynomial fitting. The curve to be differentiated is $y = \sin(t)$, with theoretical derivative $\dot{y} = \cos(t)$ (blue dash line). The derivative at time point $t = t_0$ is estimated from the noisy data with $\sigma_0 = 0.1$ by fitting a 2-nd order polynomial locally around 8 neighbors, i.e. by FCDS(8,2) (red dash line) and by Euler's scheme (green dash line)

One can calculate the truncation error of this scheme by Taylor expansion, which is $O(h^2)$.
By checking the variance of the noise, we can show that $\lambda > 0$ makes the finite difference smoother.

$$Var\left(\frac{\varepsilon_1 - \varepsilon_{-1}}{2h + \lambda/h}\right) = \frac{1}{2h^2 + \frac{\lambda^2}{2h^2} + 2\lambda}\sigma^2 < \frac{1}{2}\frac{\sigma^2}{h^2}$$

More examples of Fitted Central Derivative Schemes (FCDS) can be found in Table 2.1. It shows that decreasing the order of polynomial $n \leq m$ will smoothen the noise while lose some accuracy, and when $n = m$, the schemes are the classical central difference schemes which however have worst performance of noise control. The reason is that those schemes are deduced from Lagrange interpolation under the assumption that the curve goes exactly through those points, which is not the case for noisy data.

Figure 2.1 is an example illustrating the derivative estimated by polynomial fitting is more robust than the Euler's scheme.

FCDS (m,n)	schemes ($\lambda = 0$)	$\frac{\sigma_D^2}{(\frac{\sigma_0}{h})^2}$
(2,2)	$\frac{f_{+1}-f_{-1}}{2h} + O(\frac{1}{6}h^2)$	0.50
(4,2)	$\frac{2f_{+2}+f_{+1}-f_{-1}-2f_{-2}}{10h} + O(\frac{17}{30}h^2)$	0.10
(4,4)	$\frac{-f_{+2}+8f_{+1}-8f_{-1}+f_{-2}}{12h} + O(\frac{1}{30}h^4)$	0.90
(6,2)	$\frac{3f_{+3}+2f_{+2}+f_{+1}-f_{-1}-2f_{-2}-3f_{-3}}{28h} + O(\frac{7}{6}h^2)$	0.04
(6,4)	$\frac{-22f_{+3}+67f_{+2}+58f_{+1}-58f_{-1}-67f_{-2}+22f_{-3}}{252h} + O(\frac{131}{630}h^4)$	0.26
(6,6)	$\frac{f_{+3}-9f_{+2}+45f_{+1}-45f_{-1}+9f_{-2}-f_{-3}}{60h} + O(\frac{1}{140}h^6)$	1.17
(8,2)	$\frac{4f_{+4}+3f_{+3}+2f_{+2}+f_{+1}-f_{-1}-2f_{-2}-3f_{-3}-4f_{-4}}{60h} + O(\frac{59}{30}h^2)$	0.02
(8,4)	$\frac{-86f_{+4}+142f_{+3}+193f_{+2}+126f_{+1}-126f_{-1}-193f_{-2}-142f_{-3}+86f_{-4}}{1188h} + O(\frac{179}{270}h^4)$	0.11
(8,6)	$\frac{254f_{+4}-1381f_{+3}+2269f_{+2}+2879f_{+1}-2879f_{-1}-2269f_{-2}+1381f_{-3}+-254f_{-4}}{8580h} + O(\frac{797}{12012}h^6)$	0.42
(8,8)	$\frac{-3f_{+4}+32f_{+3}-168f_{+2}+672f_{+1}-672f_{-1}+168f_{-2}-32f_{-3}+3f_{-4}}{840h} + O(\frac{1}{630}h^8)$	1.36

Table 2.1: Fitted Central Derivative Schemes (FCDS) upto m=8,n=8

Chapter 3

Application to large scale GRN inference problem

3.1 The Data

Time-series data of gene expression datasets will be put in use:

The time-series dataset can be recorded in a bunch of matrices $\mathbf{X}_r \in \mathbb{R}^{T \times n}$, $r = 1, ..., R$. In each time-series matrix, each row is the gene expression level at different time and each column represents a given gene; different time-series matrices are the time-course data under different initial values or initial perturbations. Figure 3.1 illustrates a time-series dataset.

The open source software GeneNetWeaver (GNW) [6, 20] is an *In silico* (numerical) simulator, containing sophisticated dynamic models of gene regulatory networks of *E.coli*[21] and *S.cerevisiae*[22], including a thermodynamical model of transcriptional regulation, mRNA and protein dynamics, being able to generate gene expression data with the same noise level as those *In Vivo* (in real biological experiments). One can also find real experimental data in online databases such as *GenExpDB*.

In this chapter, all data are generated from GeneNetWeaver (GNW).

3.2 Evaluation metrics

We introduce some basic evaluation metrics used in this work. A prediction of the existence of an arc in the network can lie in four categories as shown in the table 3.1.

replicate 1

time	gene1	gene2	gene3	gene4	gene5	gene6	gene7	gene8
0	0.209	0.018	0.135	0.425	0.117	0.759	0.664	0.153
50min	0.490	0.590	0.784	0.810	0.933	0.037	0.520	0.763
100min	0.030	0.346	0.331	0.741	0.655	0.925	0.066	0.111
150min	0.130	0.845	0.356	0.532	0.275	0.148	0.724	0.860
200min	0.243	0.221	0.167	0.481	0.731	0.665	0.758	0.404
250min	0.120	0.928	0.578	0.829	0.271	0.256	0.781	0.816
300min	0.575	0.527	0.390	0.429	0.595	0.844	0.404	0.062
350min	0.003	0.553	0.365	0.053	0.375	0.857	0.010	0.835
400min	0.569	0.629	0.128	0.318	0.890	0.271	0.428	0.286
450min	0.611	0.583	0.120	0.367	0.508	0.933	0.064	0.065
500min	0.244	0.434	0.745	0.102	0.067	0.019	0.332	0.475
550min	0.151	0.488	0.241	0.225	0.519	0.352	0.600	0.714
600min	0.462	0.203	0.668	0.137	0.003	0.592	0.111	0.100
650min	0.259	0.371	0.016	0.322	0.557	0.509	0.765	0.395
700min	0.723	0.389	0.663	0.849	0.386	0.037	0.216	0.413
750min	0.891	0.064	0.573	0.303	0.065	0.475	0.109	0.135
800min	0.858	0.589	0.551	0.691	0.540	0.171	0.592	0.946
850min	0.032	0.422	0.557	0.018	0.574	0.257	0.838	0.699
900min	0.118	0.976	0.485	0.966	0.794	0.456	0.034	0.698
950min	0.246	0.375	0.189	0.204	0.463	0.850	0.842	0.224
1000min	0.268	0.209	0.480	0.826	0.289	0.297	0.773	0.428
1050min	0.879	0.355	0.902	0.652	0.587	0.197	0.864	0.613
1100min	0.254	0.136	0.565	0.073	0.406	0.731	0.610	0.951

replicate 2

time	gene1	gene2	gene3	gene4	gene5	gene6	gene7	gene8
0	0.886	0.224	0.253	0.484	0.990	0.869	0.675	0.715
50min	0.669	0.176	0.449	0.701	0.235	0.860	0.171	0.181
100min	0.566	0.221	0.962	0.416	0.614	0.069	0.645	0.487
150min	0.141	0.374	0.649	0.868	0.502	0.979	0.898	0.675
200min	0.899	0.140	0.092	0.201	0.008	0.090	0.102	0.848
250min	0.832	0.425	0.150	0.542	0.731	0.207	0.606	0.651
300min	0.889	0.698	0.912	0.430	0.622	0.749	0.153	0.576
350min	0.757	0.201	0.863	0.770	0.398	0.917	0.546	0.341
400min	0.379	0.281	0.566	0.541	0.485	0.200	0.896	0.192
450min	0.419	0.802	0.520	0.965	0.935	0.344	0.042	0.970
500min	0.648	0.185	0.827	0.981	0.688	0.116	0.160	0.055
550min	0.603	0.608	0.216	0.328	0.210	0.148	0.788	0.383
600min	0.252	0.359	0.084	0.217	0.214	0.151	0.222	0.588
650min	0.819	0.727	0.273	0.873	0.467	0.451	0.735	0.129
700min	0.435	0.858	0.307	0.448	0.955	0.709	0.397	0.792
750min	0.039	0.209	0.305	0.562	0.717	0.574	0.010	0.709
800min	0.432	0.011	0.164	0.971	0.348	0.543	0.740	0.608
850min	0.245	0.762	0.073	0.563	0.388	0.951	0.733	0.181
900min	0.474	0.273	0.094	0.702	0.501	0.078	0.710	0.878
950min	0.759	0.975	0.374	0.198	0.275	0.126	0.425	0.832
1000min	0.350	0.795	0.455	0.690	0.730	0.881	0.292	0.504
1050min	0.226	0.148	0.208	0.482	0.430	0.948	0.874	0.237
1100min	0.061	0.080	0.257	0.643	0.770	0.270	0.343	0.801

Figure 3.1: Example. A series of replicates of time-series data

in reality / in prediction	existence of an arc	non-existence of an arc
existence of an arc	True Positive	False Positive
non-existence of an arc	False Negative	True Negative

Table 3.1: Four situations a prediction could be in

True Positive (TP): the existence of an arc is predicted as positive and the prediction is correct;
False Positive (FP): the existence of an arc is predicted as positive and the prediction is incorrect;
True Negative (TN): the existence of an arc is predicted as negative and the prediction is correct;
False Negative (FN): the existence of an arc is predicted as negative and the prediction is incorrect.

Precision:

$$Precision = \frac{TP}{TP + FP}$$

Accuracy:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Recall (True Positive Rate):

$$Recall = TPR = \frac{TP}{TP + FN}$$

False Positive Rate:

$$FPR = \frac{FP}{FP + TN}$$

Prediction list:
An example is shown in Figure 3.2. An arc is presented by its source and target and the list is ranked by the scores or probability calculated in the process of prediction.

Precision-Recall Curve:
Taking first k edges in the prediction list, the pair $(Recall, Precision)_k,\ k = 1, 2, ...$ can be calculated and plotted as a curve. The Precision-Recall Curve in Figure 3.3a shows that as k increases, *Recall* increases, yet *Precision* decreases. In such a way, it tells us up to what point we should trust the prediction. The area under the Precision-Recall Curve can be written as AUPR; a larger AUPR indicates a better

prediction of arcs in a network
(ranked by probability)

source	target	score
gene3	gene1	0.860
gene4	gene3	0.666
gene1	gene2	0.639
gene2	gene1	0.474
gene3	gene4	0.469
gene2	gene3	0.449
gene1	gene3	0.361
gene4	gene1	0.315
gene4	gene2	0.283
gene1	gene4	0.240
gene3	gene2	0.196
gene2	gene4	0.077

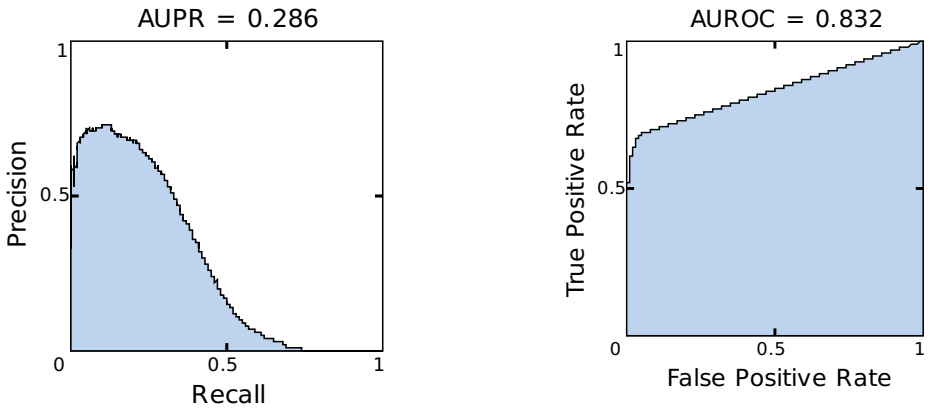
Figure 3.2: An example of prediction of arcs in a network. The probability or scores of existence of arcs are ranked in descending order.

prediction.

Receiver-Operating Characteristic (ROC) Curve:
Taking first k edges in the ranked list of predictions, the pair $(FPR, TPR)_k$, $k = 1, 2, ...$ can be calculated and plotted as a curve. The area under the Receiver-Operating Characteristic (ROC) Curve is written as AUROC; a more upwards convex ROC curve means a better performance and a 0.5 AUROC (a diagonal line) indicates the prediction is no better than random guessing; Figure 3.3b.

p-value
The p-value is calculated under the null hypothesis that the obtained results (AUROC, AUPR etc.) are merely random.

$$p\text{-}value = P(RESULT \geq obtained\ results \mid \text{prediction is random})$$



(a) AUPR example

(b) AUROC example

Figure 3.3: examples of Precision-Recall Curve and Receiver-Operating Characteristic (ROC) Curve

3.3 Application of Parameter Optimization Solver

The time-series data $\mathbf{X}_r$ of gene expression level can be directly used for parameter optimization described in Section 2.2.1 and Section 2.2.2. The estimation of derivatives in the ODEs model follows Section 2.3.

The absolute value of the parameters can be used as scores ranking the confidence of the prediction of arcs; a larger a_{ij} indicated a stronger influence from gene-j to gene-i. The sign of the parameters tells whether the interaction is an inhibition or an activation.

3.4 Results

In this section, we show the performance of the parameters identification method proposed in this thesis. The metrics for evaluating the performance are Area Under the Precision-Recall Curve (AUPR) and Area Under the Receiver-Operating Characteristic (AUROC), which have been introduced above.

The methods were tested on an *Escherichia coli(E.coli)* transcriptional network with 1565 genes (nodes) and 3758 interactions (arcs), Figure 3.4. The data for the test were generated by *GeneNetWeaver (GNW) 3.1.1 Beta*. In total, 3600 time-course replicates were generated and each time-course replicate experiment contains 21 time points ranging from 0 to 1000 with time step 50. In the last section in this chapter, we show the computation time of identification of large networks up to 10,000 nodes.

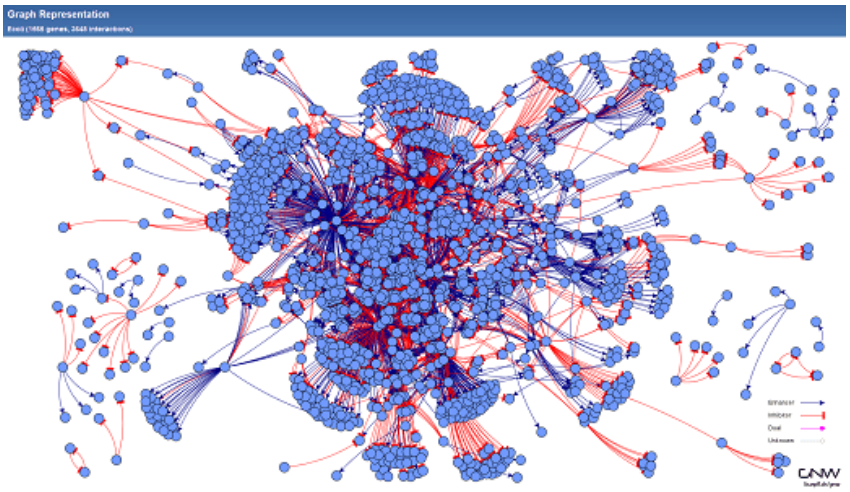


Figure 3.4: *Escherichia coli(E.coli)* transcriptional network with 1565 genes and 3758 interactions; produce by the software GWN

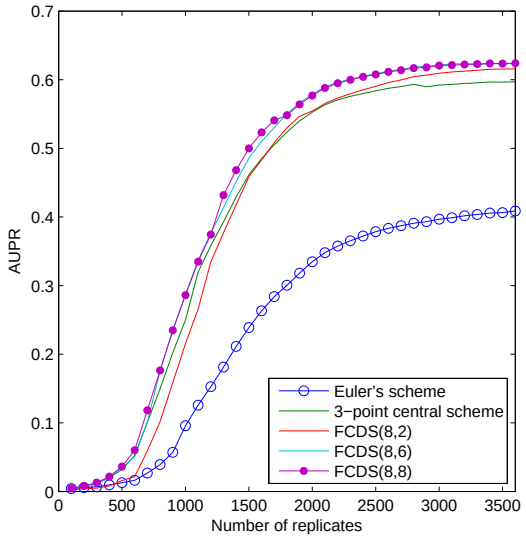
3.4.1 Reconstruction of 1565-node *E.coli* GRN

Impact of data size

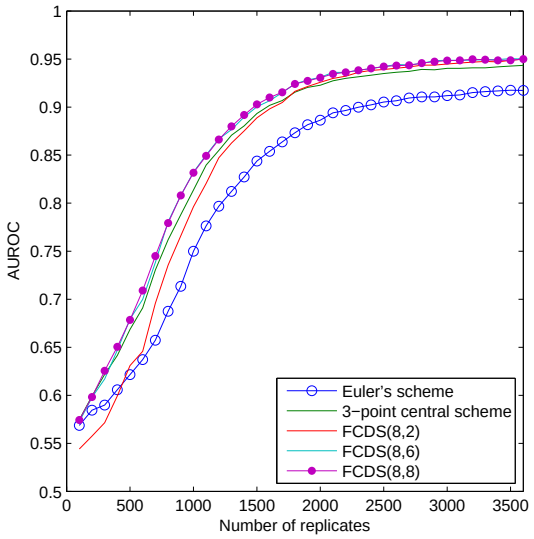
We tested the impact of data size on the performance with the 1565-node *E.coli* network. In Figure 3.5, it shows that increase of replicates of experiment leads an S-shape curve of both AUPR and AUROC. It indicates the fitting problem always requires enough data; with more data, one can expect better performance.

However, the performance has an upper limitation despite the fact that surplus data are supplied. In this test, the upper limitation of AUPR and AUROC for FCDS(8,8) scheme are 0.6241 and 0.9500 respectively. Figure 3.6b shows the Precision-Recall Curve and Receiver-Operating Characteristic (ROC) Curve when this limitation is achieved; Figure 3.6a shows when there is no enough data, the prediction is no better than merely random guessing.

The Euler's scheme, 3-point central scheme, FCDS(8,2), FCDS(8,6), and FCDS(8,8) schemes have truncation errors in the order of $O(h)$, $O(h^2)$, $O(h^2)$, $O(h^6)$, $O(h^8)$ respectively. Figure 3.5 shows that a higher order of truncation error leads to a better performance. In details, Euler's scheme ($O(h)$) had the worst performance. FCDS(8,2) and 3-point central scheme (both with $O(h^2)$) achieved similar performance in some range, but FCDS(8,2) eventually surpassed the latter. The performance of FCDS(8,6) and FCDS(8,8) went very closely to each other, which indicates truncation error of $O(h^6)$ is sufficient for this test and an increase of accuracy yields no improvement.

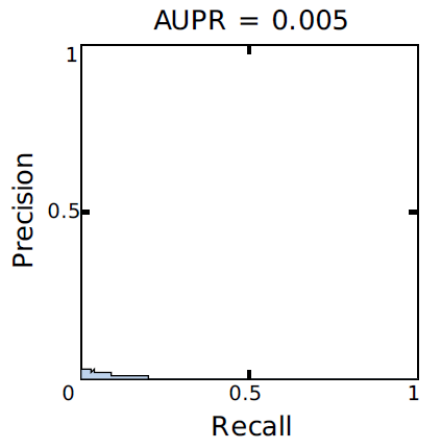


(a) AUPR v.s. data size

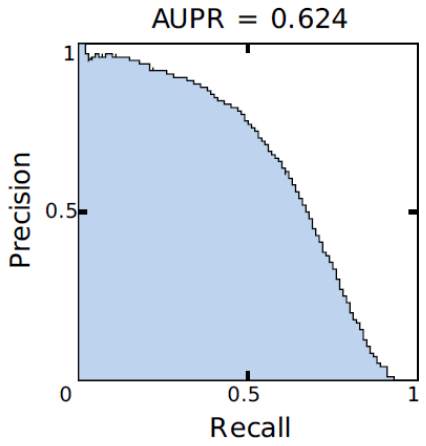


(b) AUROC v.s. data size

Figure 3.5: performance of ODEs model without pipeline on different data size, with derivatives estimated by Euler's scheme, 3-point central scheme, FCDS(8,2), FCDS(8,6), and FCDS(8,8) schemes.



(a) performance on data size = 300



(b) performance on data size = 3600

Figure 3.6: Precision-Recall Curve and ROC Curve of the prediction of ODEs model with FCDS(8,8) scheme on data size 3600 and 300 respectively.

Impact of noise level

In this section we still show the test on the 1565-node *E.coli* network. We examine how noise level (i.e. the standard deviation σ_0 of the noise) of the given data will affect the performance given adequate data and thus show noise robustness of different finite difference schemes.

As discussed in Section 2.3, the Euler's scheme, 3-point central scheme, FCDS(8,2), FCDS(8,6), and FCDS(8,8) schemes amplify the noise of the estimated derivatives to the extent of $2\frac{\sigma_0^2}{h^2}$, $0.50\frac{\sigma_0^2}{h^2}$, $0.02\frac{\sigma_0^2}{h^2}$, $0.42\frac{\sigma_0^2}{h^2}$, $1.36\frac{\sigma_0^2}{h^2}$ respectively. Figure 3.7 shows the performances of these five schemes on increasing noise level; the AUPR and AUROC of all examined schemes decreased when noise level increased; Euler's scheme caused a most rapid decline, while FCDS(8,2) gave out the best robustness if noise

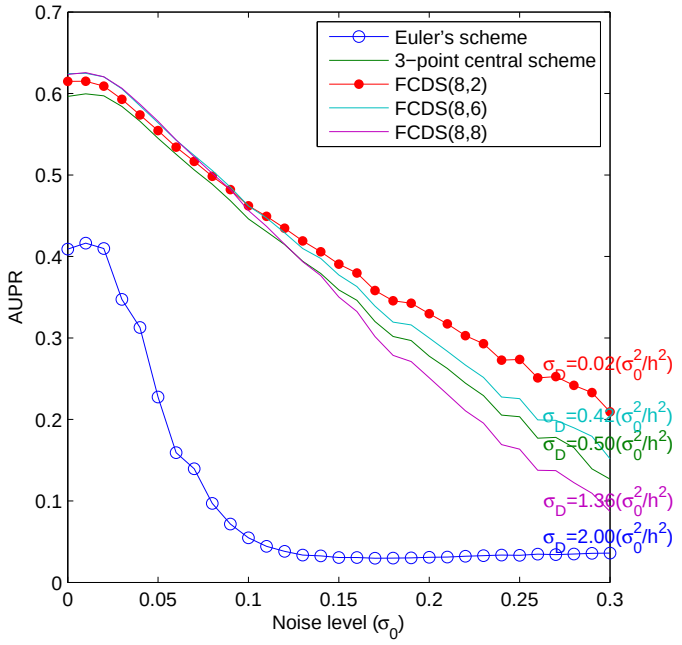
is high and its performance surpassed FCDS(8,6) around $\sigma_0 = 0.1$.

3.4.2 Computation Time

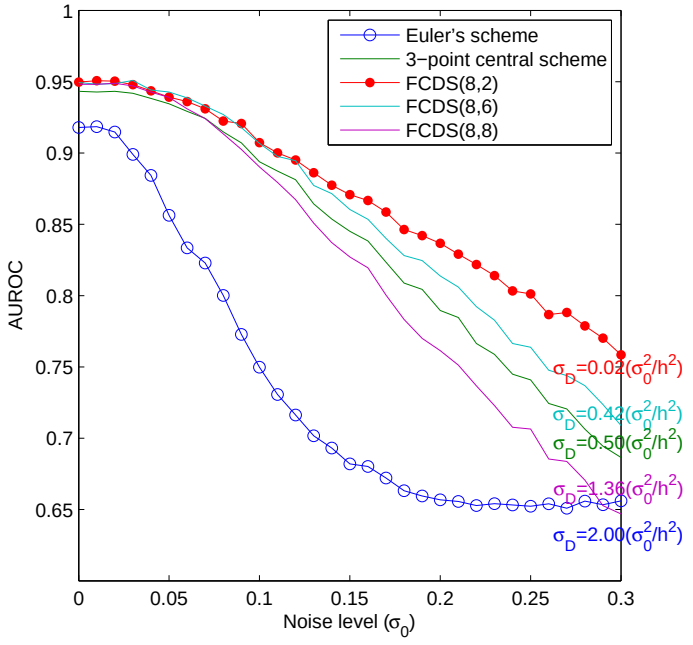
We tested the computation time with CPU *Intel Core i7* on reconstructing the networks of size ranging from n = 500 genes to n = 10,000 genes, as shown in Figure 3.8. The time for reconstruction of 500-gene network was 0.6s and for 10,000-gene network 1766s (about 30min), in which over 100 million parameters were optimized; the $(\log_{10} n, \log_{10} t)$ was fitted into a straight line which shows the time complexity of this algorithm is about $O(1/10^{7.6} \cdot n^{2.7})$.

The distribution of the computation time is shown in Figure 3.9. The linear system was solved by the Matlab backslash (\ ) operator, which is quite efficient and stable. The cross validation has to solve up to 20 times of the linear system and to compute the Frobenius norm 20 times to choose a better regularization term α in equation 2.7. Therefore, the cross validation took much time, but it could have been parallelized easily to reduce the computation time.

Note that the networks and data for this test were randomly generated, since generating the dynamic data of large networks is another challenging work and we are concerning the computation time but rather the performance in this section.

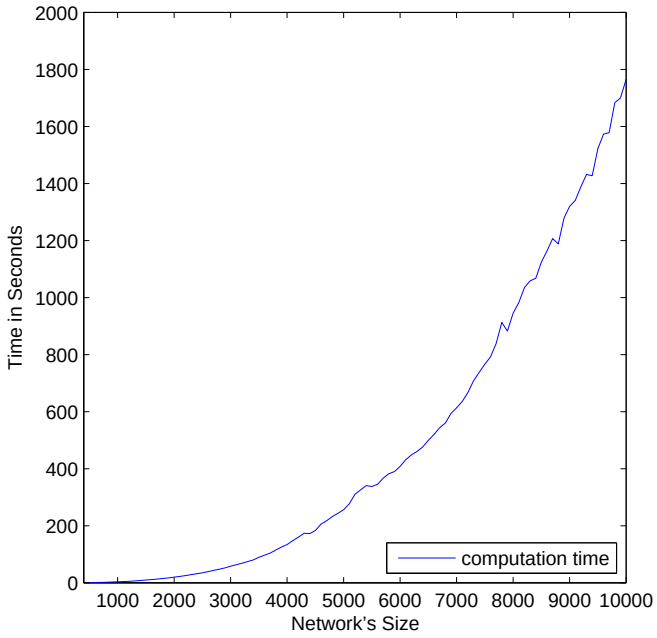


(a) AUPR v.s. noise level

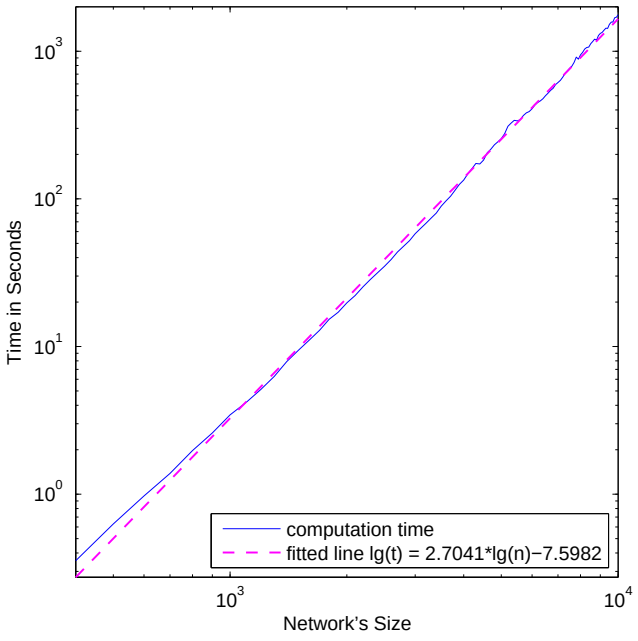


(b) AUROC v.s. noise level

Figure 3.7: performance of ODEs model on data with different noise level, with derivatives estimated by Euler's scheme, 3-point central scheme, FCDS(8,2), FCDS(8,6), and FCDS(8,8) schemes.



(a) Computation time v.s. network size



(b) Log-Log plot of computation time v.s. network size

Figure 3.8: Computation time v.s. network size.

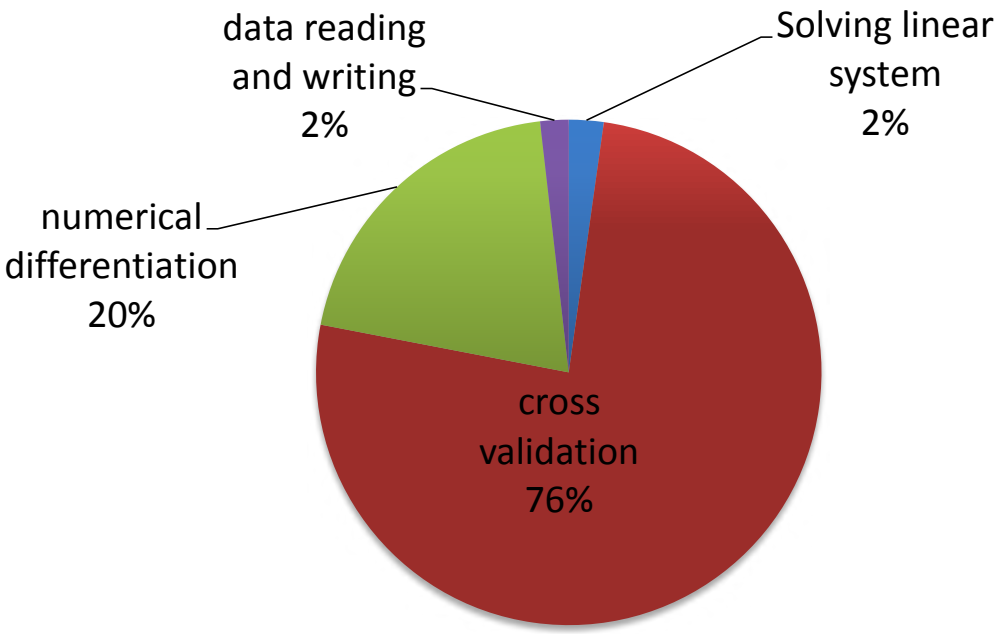


Figure 3.9: Distribution of computation time.

Chapter 4

Pipeline: pre-filtration and post-modelling

Madar etc.[9] have proposed a combination of several methods for GRN inference and named it a pipeline, which showed an improvement of performance. A combination of a sequence of algorithms is called a *pipeline*[23]. Like a line of pipes with pumps, valves and control devices for conveying liquids and gases, this pipeline of algorithms feeds the output of the precedent algorithm into the next, controlling parameters of each algorithm, giving the next algorithm a good starting point, expecting an improved final outcome.

The GRN inference problem often suffers from shortage of available data. In order to overcome this drawback, in this chapter, we introduce a filtration process before the parameter optimization step and a modelling process after that. The workflow of the pipeline proposed in this thesis is shown in Figure 4.1; the outlier detection firstly filters out unlikely arcs and thus some parameters in the ODEs are restricted to zero. The constrained optimization solution for identification of parameters described in Section 2.2.2 can then be applied. Thereafter, the knockout experiments can be simulated numerically. Unlike the steady-state knockout data, the ODE model can simulate the effects of knockout in an arbitrarily short time period. With the simulated transient knock-out data, Z-scores can be computed and normalized as final scores ranking the possible arcs.

The original method without filtration suffers shortage of data. The filtration reduces the requirement of data size and the post-modelling produces more numerical data, which can both largely complement the shortage of data. The details of each process in the pipeline will be discussed in the following sections.

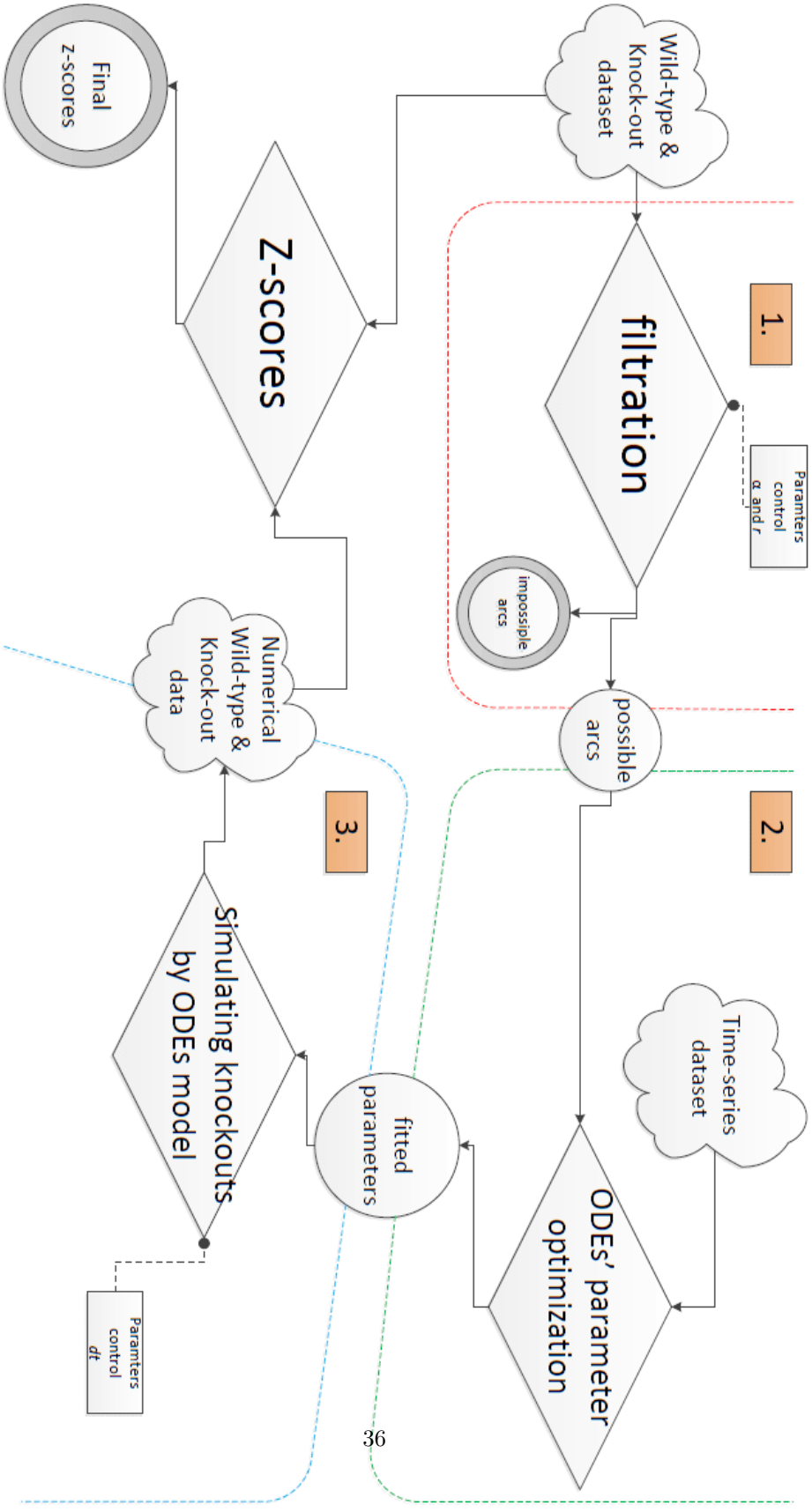


Figure 4.1: Pipeline. A combination of pre-filtration, parameter identification and post-modelling.

4.1 The Data

Three types of gene expression datasets will be put in use: *wild-type dataset, time-series dataset and steady-state knock-out dataset.*

- The wild-type dataset records the expression level of each gene in the considered network under steady state without any perturbation; it can be represented as a vector $\mathbf{x}^{wt} \in \mathbb{R}^n$ with each element x_i^{wt} the wild-type expression level of gene-i.
- The time-series dataset can be recorded in a bunch of matrices $\mathbf{X}_r \in \mathbb{R}^{T \times n}$, $r = 1, ..., R$. In each time-series matrix, each row is the gene expression level at different time and each column represents a given gene; different time-series matrices are the time-course data under different initial values or initial perturbations. A gene could be knocked out (made non-functional) by certain genetic techniques.
- The steady-state knock-out data are obtained by a series of experiments in each of which one gene is made defective and the corresponding steady states of all genes in the network are recorded in a vector; by knocking out different genes, a series of knock-out data vectors are obtained and are put together into a matrix $\mathbf{X}^{ko} = [\mathbf{x}^{ko_1}, ..., \mathbf{x}^{ko_i}, ..., \mathbf{x}^{ko_n}]$, with the entry $x_j^{ko_i}$ indicating the steady state of gene-j given gene-i is knocked out.

Figure 4.2 illustrates wild-type data and steady-state knock-out data; Figure 3.1 shows time-series data.

4.2 Filtration

4.2.1 Outlier Detection

An outlier is a data point that deviates with statistical significance from other observations [24]. An outlier can be detected graphically by plotting the data points or by statistical tests[25]. Assuming a normal distribution, Grubbs' Test[24][26] detects a single outlier and the Tietjen-Moore Test[27], a generalized Grubbs' Test, can find specified number of outliers. The Generalized Extreme Studentized Deviate (ESD) test[28] detects multiple outliers with no requirement to specify how many outliers exist, but only a upper bound of the number of outliers. The statistical tests give out a binary results of a data point to be or not to be an outlier; Z-scores will rank the outliers from the most likely to the least.

Under the assumption that a network system is typically sparse, when a certain

wild-type data

gene1	gene2	gene3	gene4	gene5	gene6	gene7	gene8
0.069	0.703	0.602	0.076	0.903	0.453	0.319	0.544

steady-state knock-out data

	gene1	gene2	gene3	gene4	gene5	gene6	gene7	gene8
gene1	0.000	0.014	0.867	0.949	0.718	0.697	0.454	0.694
gene2	0.142	0.000	0.006	0.461	0.938	0.208	0.052	0.576
gene3	0.024	0.498	0.000	0.297	0.534	0.539	0.078	0.158
gene4	0.612	0.357	0.617	0.000	0.395	0.207	0.189	0.406
gene5	0.902	0.477	0.109	0.401	0.000	0.861	0.156	0.299
gene6	0.864	0.056	0.186	0.420	0.919	0.000	0.177	0.180
gene7	0.653	0.460	0.694	0.630	0.532	0.722	0.000	0.981
gene8	0.330	0.835	0.260	0.114	0.068	0.751	0.502	0.000

Figure 4.2: Three types of data sets used in the proposed method.

state variable x_i in $\mathbf{x}$ is perturbed, most of the rest are unlikely to be significantly affected by the perturbation of x_i; hence the observed deviations between the perturbed data and wild-type data are mostly experimental errors which follow *i.i.d normal distribution*. Those state variables that are significantly affected by the perturbation of x_i can be regarded as outliers of this normal distribution, and are predicted to have interacts with x_i, and the parameters $a_{\cdot i}$ could not be zeros.

4.2.2 Generalized ESD test

The Generalized Extreme Studentized Deviate (ESD) test is a hypothesis test to detect multiple outliers up to a prescribed upper bound r in a data set that follows an approximately normal distribution, assuming $x_i \sim \mathcal{N}(\mu, \sigma^2)$.

- Hypothesis:

  $\mathbf{H}_0$: There are no ouliers in the data set;
  $\mathbf{H}_1$: There are up to r outliers in the data set.

- Test statistics: $R_1, R_2, ..., R_r$

For k from 1 to r, compute:

$$\begin{aligned} R_k &= \frac{\max_{x_i \in D_k} |x_i - \bar{x}_k|}{s_k} \\ x_m &= \arg \max_{x_i \in D_k} |x_i - \bar{x}_k| \\ D_{k+1} &= D_k \backslash \{x_m\} \end{aligned}$$

where D_1 is the original data set to be tested; $\bar{x}_k$ and s_k are sample mean and sample standard deviation of data in D_k, respectively.

- Rejection regions: $\lambda_1, \lambda_2, ..., \lambda_r$; if $R_k > \lambda_k$, reject $\mathbf{H}_0$

$$\begin{aligned} \lambda_k &= \frac{(n_k - k) t_{p_k, n_k - k - 1}}{\sqrt{(n_k - k - 1 + t^2_{p_k, n_k - k - 1})(n_k - k - 1)}} \qquad \forall k = 1, ..., r \\ p_k &= 1 - \frac{\alpha}{2(n_k - k - 1)} \end{aligned}$$

where $n_k = |D_k|$; $t_{p_k, n_k - k - 1}$ is the percentile of the $t - distribution$; α is the significance level.

- Number of outliers.
The number of outliers $n_o \leq r$ is determined by the largest k such that $R_k > \lambda_k$:

$$n_o = \max_{k=1,...,r} \{k \mid R_k > \lambda_k\}$$

There are two parameters to be specified in this method, the upper bound of number of outliers r and significance level α.

4.2.3 Modified Z-scores

The Z-score method is based on the large sample theory and normal distribution property that if $X \sim \mathcal{N}(\mu, \sigma^2)$, then $Z = \frac{(X - \mu)}{\sigma} \sim \mathcal{N}(0, 1)$. The Z-scores of a series of data point $x_1, ..., x_n \in D$ can be computed as:

$$Z_i = \frac{x_i - \bar{x}}{s}$$

where $\bar{x}$ and s are sample mean and sample standard deviation respectively, estimating μ and σ respectively.

However, when considering outliers, the sample mean and sample standard deviation are not robust and are sensitive to outliers. Instead of using $\bar{x}$ and s to approximate μ and σ, the modified Z-scores choose *sample median* $\tilde{x}$ and *Median Absolute Deviation (MAD)*:

$$MAD = \underset{x_i \in D}{\text{median}}(|x_i - \tilde{x}|),$$

and the modified Z-scores is computed as

$$\tilde{Z}_i = \frac{x_i - \tilde{x}}{1.4826 \cdot MAD} \tag{4.1}$$

Remark 6.

- There is a constant because $\hat{\sigma} = \left(\frac{1}{\Phi^{-1}(3/4)}\right) MAD \approx 1.4826 MAD$ is an unbiased estimation to the normal distribution standard deviation σ, i.e. $E(\sigma) = E(\hat{\sigma})$. ($\Phi$ is the cumulative distribution function of the standard normal distribution.).

4.2.4 Application of Outlier Detection and Z-scores

We explain how to filter out unlikely arcs in a gene regulatory network by analyzing wild-type data and knock-out data, so that some parameters in the ODEs can be restricted to zero, the dimension of the searching space can be reduced and data size required for fitting is diminished.

Under the assumption that the biological network is typically sparse[8][4], when a certain gene is knocked out, most of the rest are unlikely to be significantly affected; hence the observed deviations between steady-state knoc-kout data and wild-type data are mostly experimental errors which follow i.i.d normal distribution with mean $\mu = 0$ and variance σ^2 unknown. Those genes that are significantly affected by knocking out of a certain gene can be regarded as outliers of this normal distribution, and are predicted to have connections with the knocked-out gene. Meanwhile, the z-scores can be computed from the deviations between steady-state knock-out data and wild-type data.

If gene-i has an arc (or just a path) leading to gene-j in topology and is knocked out in the experiment, it would be expected that the knock-out steady-state expression level of gene-j, denoted as $x_j^{ko_i}$, changes significantly from its wide-type level x_j^{wt} such that the Generalized ESD test is able to detect the deviation $x_j^{ko_i} - x_j^{wt}$ as an outlier and the modified Z-score will also be high.

If gene-i has not so much direct or indirect interaction with gene-j in topology and is knocked out in the experiment, it would be expected that $x_j^{ko_i}$ does not change significantly besides noisy fluctuation from its wide-type level x_j^{wt} such that the Generalized ESD test is not able to detect the deviation $x_j^{ko_i} - x_j^{wt}$ as an outlier and the modified Z-score will not be high.

There are two parameters to be specified in the filtration process as mentioned in Section 4.2.1, the upper bound of number of outliers r and significance level α.

In the context of network, the upper bound of number of outliers r is the limit of in-degrees of all gene nodes. Since only up to r outliers will be detected, only up to r regulators to each gene can be found. One can choose r basing on the size of the network to be reconstructed, prior knowledge, personal experience or subjective expectations. One should also consider the amount of the available data sets, since a larger r yields more non-zero parameters in ODE, which then will require more data to do fitting.

Significance level α is the probability of *type I error* that null hypothesis is rejected when actually true. Another type of error *(type II error)* could occur when null hypothesis is accepted when it is actually false. In the context of our topic, they can be interpreted as:

$$\alpha = P(\text{type I error}) = P(\text{an arc is detected when there is actually no arc})$$

$$\beta = P(\text{type II error}) = P(\text{an arc is not detected but there is actually an arc})$$

Type I and type II errors are related. A decrease of one will always results in an increase of the other, given that sample size stays unchanged[29].

It is very important to be aware of those two types of errors. When using the method alone to detect arcs (outliers), one should chose a small significance level α so that the type I error is small. However, when applying to filtration, it is more important to have a small type II error. The reason is that once an arc is falsely filtered out, which is more likely to happen when β is large, it will never be rediscovered by following algorithms, because the corresponding parameter in ODE model will have been set to zero; on the contrary, if type I error is large, the falsely accepted link may still be detected as negative by following algorithms. Therefore, one should choose a large significance level α so that the type II error is small.

Tips

- Choose a **large** significance level α when the outlier detection technique is applied to filtration.
- Choose a **small** significance level α when the outlier detection technique is applied to find possible interactions by itself..

4.3 Simulation of Knock-out Experiment

It has been given the linear ODEs model in equation (1.1.2). The knock-out experiment can be conveniently simulated by setting the corresponding parameters to zero. For instance, a gene-j knock-out experiment can be done by setting:

$$a_{ij} = 0 \ \forall i = 1, ..., n \text{ and } i \neq j;$$
$$a_{jk} = 0 \ \forall k = 1, ..., n \text{ and } k \neq j.$$

The new parameter matrix after gene-j has been knocked out can be denoted as $\mathbf{A}_{ko_j}$, and the ODE becomes:

$$\frac{d\mathbf{x}}{dt} = \mathbf{a_0} + \mathbf{A}_{ko_j}\mathbf{x},$$

of which the theoretical solution is:

$$\mathbf{x}^{ko_j}(t) = \left(\mathbf{x}(t_0) + \mathbf{A}_{ko_j}^{-1}\mathbf{a_0}\right) e^{\mathbf{A}_{ko_j}(t-t_0)} - \mathbf{A}_{ko_j}^{-1}\mathbf{a_0} \tag{4.2}$$

Remark 7.

- Equation (4.2) gives the expression level of all genes at time t, given gene-j is knocked out. One can see the steady-states expression level of all genes are

  $$\mathbf{x}^{ko_j}(t \to \infty) = -\mathbf{A}_{ko_j}^{-1}\mathbf{a_0},$$

  given gene-j is knocked out.

- The network inference method based on steady-state knock-out experiment and any other methods based on steady-state data have a drawback that they usually have bad performance in distinguishing direct link (an arc) from indirect link (a path) in the network, because the knock-out perturbation has been spread far away into the network when the steady state is established.

- This drawback of steady-state knock-out data can be overcome by a transient numerical knock-out experiment with a small Δt:

$$\mathbf{x}^{ko_j}(t_0 + \Delta t) = \left(\mathbf{x}(t_0) + \mathbf{A}_{ko_j}^{-1}\mathbf{a_0}\right) e^{\mathbf{A}_{ko_j}\Delta t} - \mathbf{A}_{ko_j}^{-1}\mathbf{a_0} \tag{4.3}$$

  With a small Δt, the knock-out perturbation only could affect its near neighbor nodes so that the indirect link could not be seen.

Thereafter, the Z-scores can be computed from the transient numerical knock-out data and used for ranking.

Chapter 5

Application to DREAM project

5.1 The DREAM project

DREAM is an acronym for for *Dialogue for Reverse Engineering Assessments and Method*. The project's organizers will post a set of challenges especially in network reconstruction problems each year. In the DREAM3(2008), DREAM4(2009), DREAM5(2010) network inference challenges, the project holder subtracted several sub-networks from real-world biological networks and generated numerical data with GeneNetWeaver (GNW). Participant teams competed with their proposed methods, under the same information, with the same available datasets, not knowing the structure of the network to be reconstructed. After the challenge closed, the performance of the participants were scored and ranked, the previously unknown network (gold standard network) has become accessible and people can analyze and evaluate their methods with the gold standard network afterwards.

Before the DREAM project, researchers have used many algorithms to recover the network's topology and evaluated their performance by various metrics. DREAM project not only provides the same datasets, but also establishes a set of evaluation metrics[30], becoming a platform on which all researchers can assess their performance under the same standard.

5.2 Performance on DREAM4 Network Challenge

The DREAM4 *in silico* network challenge was held at the Broad Institute of MIT and Harvard in 2009. There were 19 teams participated in the *InSilico_Size100* sub-challenge,and their ranks of performance have been available online. This sub-

challenge contains five networks of size 100 to be reconstructed and the wild-type data set, knock-out data set and 10 replicates of time-series data set.

Since only 10 replicates of time-series data are provided which is insufficient to determine 10,100 parameters for a 100-node network if ODE model is applied alone, the pipeline becomes a better choice in which a pre-filtration procedure can reduce dramatically the searching dimension. The effectiveness of filtration on *network-1* in the DREAM4 challenge is shown in Figure 5.1; the original 9900 possible arcs in a 100-gene network was reduced to a few hundreds, so that the required data size for fitting was dramatically reduced.

We firstly display the effect of choosing the significance level α in the pre-filtration procedure, then we compare the performance of the presented pipeline method with those of participated teams, using the same data provided by the challenge organizers.

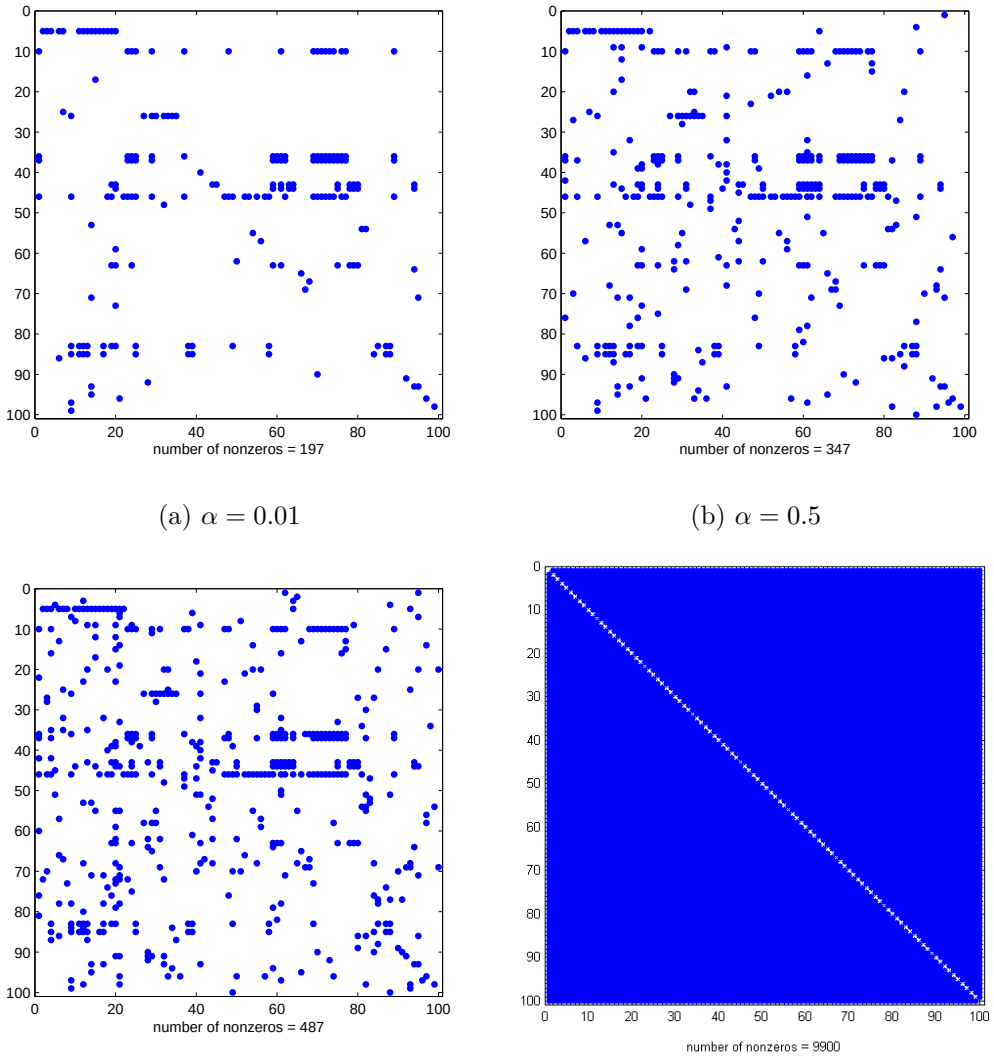
Figure 5.2 shows both the precision $TP/(TP + FP)$ and the False Negative Rate $FN/(FN + TP)$ decreased as α increased in the prediction of *Network 1*. A pre-filtration is good if the False Negative Rate is low, while a prediction is bad if the precision is low; it is consistent with the statement in Section 4.2.4 that a large α is recommended in the pre-filtration while a small α in separate use. Table 5.2 shows the performance on the five networks was improved by increasing the significance level α.

Now, we compare the performance with those of participated teams. The rank list of the 19 participants is shown in Figure 5.3, in which Team 395 ranked first with overall score 71.589, and also topped in the sub-ranking of AUPR with score 103.068; Team 548 ranked at top in the AUROC with score 40.962. Table 5.2 shows the performance of the proposed method on the five 100-gene networks and the comparison with the top performers in the challenge.

5.3 More about the DREAM4 Challenge

We have stated that the filtration process has errors; Figure 5.2. In this section, we would like to explore the potential of the linear ODEs model without filtration, given adequate data are available.

The software *GeneNetWeaver (GNW) 3.1.1 Beta* can be set to generate the time-series data for the five networks in DREAM4 Challenge with the same model and same noise level as those in the challenge, so that it allows us to produce as many data as we want and show how the performance is improved with more data, as in Table 5.3. On one hand, the method with filtration has given a quite good result especially with such little data; one the other hand, if filtration is not applied,



(a) $\alpha = 0.01$

(b) $\alpha = 0.5$

(c) $\alpha = 1$

(d) before filtration

Figure 5.1: Effectiveness of filtration with different α, with upper in-degree bound $r = 20$

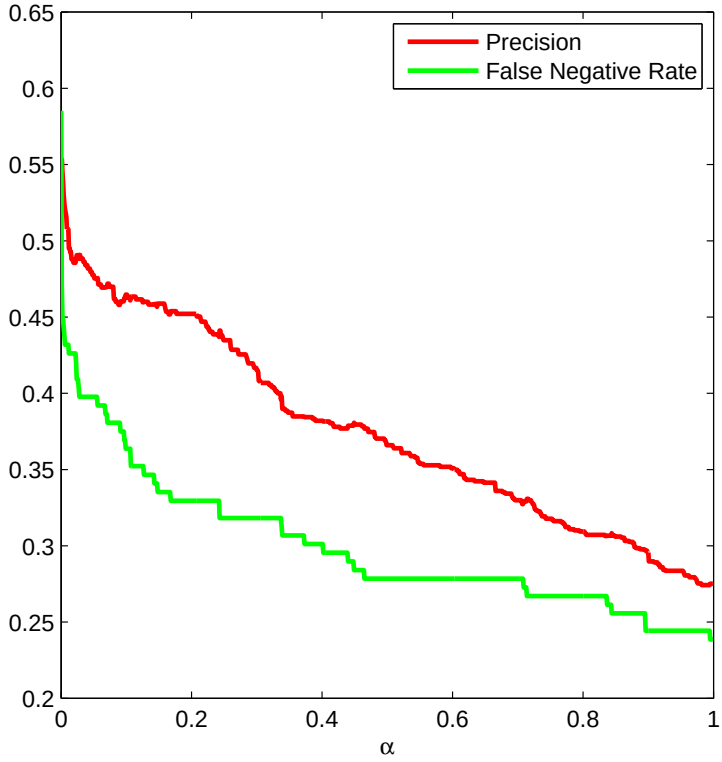


Figure 5.2: Presicion and False Negative Rate (FNR) on the network 1 in the DREAM4 challenge v.s. significance level α. The precision goes downward as α increases, indicating an increasing *type I error*; a downwards going FNR indicates a decreasing *type II error*.

with sufficient data the method can achieve even a better performance despite the fact that the amount of data required is so large. One can also see how poor the performance would be if the method is applied on a small data set without filtration.

Overall Scores

Team	OVERALL SCORE	AUROC SCORE	AUPR SCORE
Team 395	71.589	40.110	103.068
Team 296	71.297	39.737	102.857
Team 515	64.715	37.324	92.106
Team 466	63.406	37.721	89.092
Team 549	63.105	39.270	86.939
Team 271	60.310	27.861	92.760
Team 532	54.491	24.772	84.211
Team 548	52.506	40.962	64.050
Team 415	51.185	30.834	71.536
Team 161	38.500	17.190	59.811
Team 524	32.762	25.767	39.757
Team 546	29.557	18.697	40.416
Team 498	23.010	14.808	31.212
Team 236	14.309	16.618	12.001
Team 538	12.308	6.983	17.633
Team 459	6.299	5.571	7.027
Team 396	2.848	2.668	3.028
Team 540	0.155	0.121	0.188
Team 522	0.011	0.018	0.005

Figure 5.3: Rank list of 19 teams participated in the *InSilico_Size100 sub-challenge* of DREAM4. (Source:http://the-dream-project.org)

		$\alpha = 0.01$	$\alpha = 0.5$	$\alpha = 0.9$
AUPR	Net1	0.571	0.627	0.630
	Net2	0.430	0.460	0.448
	Net3	0.316	0.407	0.413
	Net4	0.389	0.467	0.491
	Net5	0.173	0.249	0.251
AUROC	Net1	0.878	0.894	0.916
	Net2	0.783	0.854	0.868
	Net3	0.769	0.813	0.797
	Net4	0.807	0.852	0.852
	Net5	0.711	0.773	0.803
overall AUPR Score		106.878	125.042	125.345
overall AUROC Score		31.969	42.378	44.250
overall Score		69.424	83.710	84.798

Table 5.1: Performance increases as significance level α increases.

Performance	AUPR	p-value of AUPR
Net1	**0.630** (Team395: 0.536)	**1.60E-150** (Team395: 1.23E-121)
Net2	**0.448** (Team296: 0.396)	**8.31E-206** (Team296: 1.80E-177)
Net3	**0.413** (Team395: 0.390)	**7.94E-101** (Team395: 5.20E-95)
Net4	**0.491** (Team271: 0.403)	**6.41E-117** (Team271: 2.93E-95)
Net5	0.251 (Team532: 0.326)	2.78E-56 (Team532: 3.82E-74)

(a) AUPR of pipeline on 5 networks respectively

Performance	AUROC	p-value of AUROC
Net1	0.916 (Team548: 0.917)	2.94E-41 (Team548: 1.92E-41)
Net2	**0.868** (Team395: 0.801)	**3.65E-64** (Team395: 4.33E-45)
Net3	0.797 (Team515: 0.844)	5.27E-39 (Team515: 2.84E-51)
Net4	**0.852** (Team549: 0.848)	**4.20E-45** (Team549: 2.56E-44)
Net5	**0.803** (Team548: 0.778)	**2.36E-35** (Team548: 1.82E-30)

(b) AUROC of pipeline on 5 networks respectively

Overall AURR score	**125.345** (Team395: 103.068)
Overall AUROC score	**44.250** (Team548: 40.962)
Overall score	**84.798** (Team395: 71.589)

(c) Overall scores on on 5 networks respectively

Table 5.2: Performance of the pipeline on 5 networks in the DREAM4 challenge respectively. The top teams in each subcategory are in the parentheses with their scores; the bold item indicates the score has exceeded the top one. The performance is achieved with these setting: $\alpha = 0.9$, $r = 20$ in pre-filtration, FCDS(8,6) scheme in ODEs model and $dt = 0.1$ in post-modelling.

Data Size		R=10	R=110	R=1110	R=2110	R*=10	R**=10
AUPR	Net1	0.168	0.496	0.605	0.612	**0.630**	0.536
	Net2	0.100	0.271	0.429	0.438	**0.448**	0.396
	Net3	0.074	0.352	0.487	**0.496**	0.413	0.390
	Net4	0.124	0.409	0.540	**0.553**	0.491	0.403
	Net5	0.046	0.293	0.434	**0.441**	0.251	0.326
AUROC	Net1	0.774	0.863	0.925	**0.935**	0.916	0.917
	Net2	0.654	0.744	0.870	**0.881**	0.868	0.801
	Net3	0.641	0.800	0.851	**0.864**	0.797	0.844
	Net4	0.720	0.820	0.890	**0.913**	0.852	0.848
	Net5	0.644	0.789	0.871	**0.881**	0.803	0.778
overall AUPR Score		22.270	94.697	136.954	**139.813**	125.345	103.068
overall AUROC Score		14.893	34.335	52.182	**55.495**	44.250	40.962
overall Score		18.581	64.516	94.568	**97.654**	84.798	71.589

Table 5.3: Performance on different data size. R is the number of replicates of time-series data. Columns under 'R' are the performances without filtration; R* represents the performances with filtration and R** represents the performances of top participants in DREAM4 Challenge, as in Table 5.2. The highest scores are emboldened.

Chapter 6

Conclusions

We focused identification of parameters on the linear ODEs system; a computationally cheap solution of both unconstrained and constrained optimization problems in Frobenius norm were provided, so that we could determine parameters in large scale linear ODEs system from experimental data. Since the experimental data are usually noisy, a series of fitted central difference schemes were provided to handle with the derivatives in the ODEs system. For sparse ODEs system, we presented the way of outlier detection to introduce sparsity into the system.

We applied the unconstrained solution to reconstruct gene regulatory networks, the dynamics of which were simplified into a linear ODEs system. The results on the 1565-gene *E.coli* regulatory networks, of which about 2.5 million parameters have been determined, showed that the proposed methods would have a satisfactory performance if sufficient data were provided, and that the noise-robust difference schemes played an important role to improve the performance. In this test, the schemes that have better noise robustness in theoretical analysis achieved better performance as the noise level increased. Therefore, the noise analysis can give us a prior knowledge of choosing a finite difference scheme before actually conducting a numerical experiment.

The constrained solution with filtration to introduce sparsity was evaluated in the DREAM4 *In Silico 100-gene network Challenge*. The filtration process showed powerful ability to reduce the number of parameters in the ODEs system from about 10,000 to about 500. With the given data in the challenge, the performance topped all participants of this challenge with an appreciable overall score. While the filtration is able to reduce the requirement of data size remarkably, it has errors: the significance level controls the effectiveness of filtration. A lower significance level filters out more parameters while has a higher risk of falsely filtering out nonzero parameters. With more data besides the given ones, the unconstrained solution without filtration even significantly raised the overall scores. There should be a trade-off between the filtration and data availability.

The computation time was evaluated by varying the size of ODEs system from 500 to 10,000 variables, which yielded over 0.25 million to 100 million parameters to be optimized and the time costed ranged from 0.6 seconds to 30 minutes. The estimated time complexity was $O(1/10^{7.6} \cdot n^{2.7})$.

Acknowledgements

I offer my gratitude to my supervisors Narsis Kiani and Hector Zenil (Unit of Computational Medicine, Karolinska Institutet, Sweden) for guiding me into the topic of biological networks, providing me advice and offering me help with their patience and trust. I give my acknowledgement to the marvelous Erasmus Mundus Program of European Union: Computer Simulations for Science and Engineering (COSSE), which has been funding my master's education during the last two years. In particular, I thank the coordinators of this program, Michael Hanke at KTH Royal Institute of Technology, Sweden and Reinhard Nabben at Technische Universität Berlin (TU Berlin), Germany for taking care of everything during my study. To my family, I am especially grateful for their emotional supports and inspiring me to follow my dreams.

Bibliography

[1] Boris Vexler. *Adaptive finite element methods for parameter identification problems.* Springer, 2013.

[2] Jim O Ramsay, G Hooker, D Campbell, and J Cao. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796, 2007.

[3] Guy Karlebach and Ron Shamir. Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9(10):770–780, 2008.

[4] Yan KK Gerstein M Yip KY, Alexander RP. Improved reconstruction of in silico gene regulatory networks by integrating knockout and perturbation data. *PLos ONE*, 2010.

[5] Florian Greil. *Dynamics of Boolean networks.* PhD thesis, TU Darmstadt, 2009.

[6] Daniel Marbach, Thomas Schaffter, Claudio Mattiussi, and Dario Floreano. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Bology*, 16(2):229–239, 2009. WingX.

[7] Madar A. Ostrer H. Bonneau R. Greenfield, A. Dream4: Combining genetic and dynamic information to identify biological networks and dynamical models. *PLos ONE*, 2010.

[8] Eric Vanden-Eijnden Richard Bonneau Aviv Madar, Alex Greenfield. Dream3: Network inference using dynamic context likelihood of relatedness and the inferelator. *PLos ONE*, 2010.

[9] Ostrer H-Vanden-Eijnden E Bonneau R Madar A, Greenfield A. The inferelator 2.0: A scalable framework for reconstruction of dynamic regulatory network models. *Engineering in Medicine and Biology Society*, 2009.

[10] George Michailidis James Henderson. Network reconstruction using nonparametric additive ode models. *PLos ONE*, 2014.

[11] Christopher Fogelberg and Vasile Palade. Machine learning and genetic regulatory networks: A review and a roadmap. In Aboul-Ella Hassanien, Ajith Abraham, AthanasiosV. Vasilakos, and Witold Pedrycz, editors, *Foundations of Computational, Intelligence Volume 1*, volume 201 of *Studies in Computational Intelligence*, pages 3–34. Springer Berlin Heidelberg, 2009.

[12] M. K. Stephen Yeung, Jesper Tegner, and James J. Collins. Reverse engineering gene networks using singular value decomposition and robust regression. *Proceedings of the National Academy of Sciences*, 99(9):6163–6168, 2002.

[13] Matthieu Vignes, Jimmy Vandel, David Allouche, Nidal Ramadan-Alban, Christine Cierco-Ayrolles, Thomas Schiex, Brigitte Mangin, and Simon de Givry. Gene regulatory network reconstruction using bayesian networks, the dantzig selector, the lasso and their meta-analysis. *PLoS ONE*, 6(12):e29165, 12 2011.

[14] G. W. Stewart. *Matrix Algorithms: Volume 1, Basic Decompositions*. Matrix Algorithms. Society for Industrial and Applied Mathematics, 1998.

[15] Rick Chartrand. Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics*, 2011.

[16] Chein-Shan Liu and Satya N Atluri. A fictitious time integration method for the numerical solution of the fredholm integral equation and for numerical differentiation of noisy data, and its relation to the filter theory. *Computer Modeling in Engineering and Sciences (CMES)*, 41(3):243, 2009.

[17] Herman J Woltring. On optimal smoothing and derivative estimation from noisy displacement data in biomechanics. *Human Movement Science*, 4(3):229–245, 1985.

[18] R.L. Burden and J.D. Faires. *Numerical Analysis*. Brooks/Cole, Cengage Learning, 2011.

[19] P. Deuflhard and F. Bornemann. *Scientific Computing with Ordinary Differential Equations*. Texts in Applied Mathematics. Springer, 2002.

[20] Thomas Schaffter, Daniel Marbach, and Dario Floreano. Genenetweaver:in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16):2263–2270, 2011. wingx.

[21] Socorro Gama-Castro, Verónica Jiménez-Jacinto, Martín Peralta-Gil, Alberto Santos-Zavaleta, Mónica I Peñaloza-Spinola, Bruno Contreras-Moreira, Juan Segura-Salazar, Luis Muñiz-Rascado, Irma Martínez-Flores, Heladia Salgado, et al. Regulondb (version 6.0): gene regulation model of escherichia coli k-12 beyond transcription, active (experimental) annotated promoters and textpresso navigation. *Nucleic acids research*, 36(suppl 1):D120–D124, 2008.

[22] S. Balaji, M. Madan Babu, Lakshminarayan M. Iyer, Nicholas M. Luscombe, and L. Aravind. Comprehensive analysis of combinatorial regulation using the transcriptional regulatory network of yeast. *Journal of Molecular Biology*, 360(1):213 – 227, 2006.

[23] J.M. Lingeman and D. Shasha. *Network Inference in Molecular Biology: A Hands-on Framework*. SpringerBriefs in Electrical and Computer Engineering. Springer, 2012.

[24] F. E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 1969.

[25] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Outlier detection techniques. In *Tutorial at the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2009.

[26] Wilhelmine Stefansky. Rejecting outliers in factorial designs. *Technometrics*, 14(2):469–479, 1972.

[27] Gary L Tietjen and Roger H Moore. Some grubbs-type statistics for the detection of several outliers. *Technometrics*, 14(3):583–597, 1972.

[28] Bernard Rosner. Percentage points for a generalized esd many-outlier procedure. *Technometrics*, 1983.

[29] D.C. Montgomery and G.C. Runger. *Applied statistics and probability for engineers*. John Wiley & Sons, 1994.

[30] Robert J. Prill, Daniel Marbach, Julio Saez-Rodriguez, Peter K. Sorger, Leonidas G. Alexopoulos, Xiaowei Xue, Neil D. Clarke, Gregoire Altan-Bonnet, and Gustavo Stolovitzky. Towards a rigorous assessment of systems biology models: The dream3 challenges. *PLoS ONE*, 5(2):e9202, 02 2010.

TRITA-MAT-E 2014:60
ISRN-KTH/MAT/E—14/60-SE

www.kth.se