UNIVERSITÉ DU LUXEMBOURG

PhD-FSTC-2018-12
The Faculty of Sciences, Technology and Communication

DISSERTATION

Presented on 21/02/2018 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN SCIENCES DE L'INGÉNIEUR

by

Zuogong YUE
Born on 18 May 1988 in Shanxi (China)

DYNAMIC NETWORK RECONSTRUCTION IN SYSTEMS BIOLOGY: METHODS AND ALGORITHMS

Declaration

I hereby declare that this dissertation has been written only by the undersigned and without any assistance from third parties. This dissertation is the result of my own work. It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Luxembourg or any other University or similar institution. I further state that no substantial part of my dissertation has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Luxembourg or any other University of similar institution.

Zuogong Yue
March 2018

Acknowledgements

I would first and foremost like to thank my supervisor Prof. Jorge Goncalves for providing me with the opportunity to work on such an interesting project in his systems control group. This work would not be possible without your guidance and advice. Your consistent interests on computational methods for biology give birth to the topics in my work. And your encouragement and supports allow to explore my potentials on independent scientific studies, with particular considerations on generic applications. I really appreciate that you give me trust and freedom to formulate and expand upon our ideas.

Scientific study never fails to benefit from collaborative effort and exchange of ideas. Thanks for my enthusiasm collaborators Dr. Johan Thunberg and Prof. Lennart Ljung. You are always ready to help when I get stuck on problems. I am so honored to have both of you as my main collaborators, whose rich knowledge and board visions guide me to the right way. Every time before the submission of papers, you were always there for review and proof reading. Moreover, I would also like to thank members in my group: Dr. Atte Aalto, Dr. Wei Pan, Dr. Rucha Sawlekar, Dr. Stefano Magni, and Dr. Andreas Husch. Thanks for your patience on my questions and your share of ideas.

Prof. Johan Schoukens, Prof. Sean Warnick, Prof. Massimiliano Esposito and Dr. Ronan Fleming, thanks for taking the time off your schedules to take part in my PhD defence. It is a great honor to have you in my jury. I also would like to thank my CET committee members Prof. Ye Yuan and Dr. Ronan Fleming, who were always supportive of my work throughout my PhD studies.

I also want to thank Fonds National de Recherche, Luxembourg, for funding my work by AFR PhD grant (9247977). I am extremely thankful for Dr. Johan Thunberg and Dr. Jasmin Sinha who helped me for the grant application. Moreover, I would like to show my appreciation to all secretaries in LCSB supporting my PhD studies.

At last, I thank my family and friends for everything! Their contribution to my life and career remains incomparable.

Abstract

Dynamic network reconstruction refers to a class of problems that explore causal interactions between variables operating in dynamical systems. This dissertation focuses on methods and algorithms that reconstruct/infer network topology or dynamics from observations of an unknown system. The essential challenges, compared to system identification, are imposing sparsity on network topology and ensuring network identifiability. This work studies the following cases: multiple experiments with heterogeneity, low sampling frequency and nonlinearity, which are generic in biology that make reconstruction problems particularly challenging.

The *heterogeneous* data sets are measurements in multiple experiments from the underlying dynamical systems that are different in parameters, whereas the network topology is assumed to be consistent. It is particularly common in biological applications. This dissertation proposes a way to deal with multiple data sets together to increase computational robustness. Furthermore, it can also be used to enforce network identifiability by multiple experiments with input perturbations.

The necessity to study low-sampling-frequency data is due to the mismatch of network topology of discrete-time and continuous-time models. It is generally assumed that the underlying physical systems are evolving over time continuously. An important concept *system aliasing* is introduced to manifest whether the continuous system can be uniquely determined from its associated discrete-time model with the specified sampling frequency. A Nyquist-Shannon-like sampling theorem is provided to determine the critical sampling frequency for system aliasing. The reconstruction method integrates the *Expectation Maximization* (EM) method with a modified *Sparse Bayesian Learning* (SBL) to deal with reconstruction from output measurements.

A tentative study on nonlinear Boolean network reconstruction is provided. The nonlinear Boolean network is considered as a union of local networks of linearized dynamical systems. The reconstruction method extends the algorithm used for heterogeneous data sets to provide an approximated inference but improve computational robustness significantly.

The reconstruction algorithms are implemented in MATLAB and wrapped as a package. With considerations on generic signal features in practice, this work contributes to practically useful network reconstruction methods in biological applications.

Table of contents

Nomenclature

Roman Symbols

$\mathbb{R}$	The set of real numbers
$\mathbb{C}$	The set of complex numbers
$\mathbb{N}$	The set of natural numbers
$\mathbb{R}_+$	The set of non-negative real numbers
$\mathrm{Re}(x)$	Real part of $x \in \mathbb{C}$
$\mathrm{Im}(x)$	Imaginary part of $x \in \mathbb{C}$
$\mathbb{E}$	Expectation
$A(i,j)$	Entry in row i and column j of matrix A
$A(i,:)$	Row i of matrix A
$A(:,j)$	Column j of matrix A
A^T	Transpose of A
A^{-1}	Inverse of A
A^{-T}	Shorthand of $(A^{-1})^T$
$\lambda(A)$	Eigenvalues of A
I	Identity matrix of implicit dimension
0	Zero matrix of implicit dimension
$\mathbf{1}$	Vector of 1's of implicit dimension
$\mathrm{card}(x)$	Cardinality of $x \in \mathbb{R}^n$

$\operatorname{diag}(X)$	A vector of diagonal elements of X if X is a matrix; a diagonal matrix with X as its diagonal elements if X is a vector
$\mathbf{dom}\, f$	Domain of function f
s	Differential operator or a complex variable in Laplace transform
z	Forward shift operator or a complex variable in the **z**-transform
q	A dummy operator denoting both s and z
(Q, P, H)	Dynamical Structure Function with transfer matrices Q, P, H
(A, B, C, D)	State-space realization with specified system matrices
(A, B)	State-space realization with $C = \begin{bmatrix} I & 0 \end{bmatrix}$ and $D = 0$

Acronyms / Abbreviations

ADMM	Alternative direction method of multipliers
ARD	Automatic Relevance Determination
ARMAX	Autoregressive moving-average models with exogenous inputs
ARX	Autoregressive models with exogenous inputs
CCCP	Convex-Concave Procedure
DCP	Disciplined convex programming
DSF	Dynamical Structure Function
EM	Expectation Maximization
GC	Granger Causality
LTI	Linear Time-Invariant
MAP	Maximum a posteriori
ML	Maximum likelihood
PEM	Prediction error minimization
QP	Quadratic program
SBL	Sparse Bayesian Learning
SSM	State space model

Chapter 1

Introduction

Network reconstruction it the process that builds path diagrams, called “networks”, from data sets. The term “network” refers to undirected or directed graphs, with or without capacity functions that define the weights associated with edges. The name comes from the legacy of biology, instead of the strict definition of networks in the graph theory or computer science. The data can be cross-section data, panel data or time series, which depend on the requirements of the methods used in network reconstruction. The problem of network reconstruction is a general inverse problem motivated by many phenomena which are naturally described as networks of interconnected systems.

Dynamic network reconstruction refers to reconstructing from time series directed graphs (with capacity functions, if necessary) that describe interconnections between measured variables, by identifying network models for the underlying dynamical systems. Dynamic networks, the results of dynamic network reconstruction, do deliver causal information. We avoid using the term “causal network reconstruction” to differentiate it from network inference/reconstruction methods based on certain widely used definitions of *causality* that are given by conditional independence in probability, which will be reviewed later. One will see overlaps between these two classes of approaches.

This dissertation studies methods and algorithms in dynamic network reconstruction, with focuses on particular issues raised by the properties of biological data. The network model used in our studies is called *Dynamical Structure Functions* (DSF), which will reviewed in detailed in Chapter 2. Specific problems considered in this dissertation are summarized in Section 1.4. Motivation to the work is given in Section 1.1, and literature review of available network reconstruction methods follows in Section 1.2. The essential preliminary tools in mathematics used in later chapters are provided in Section 1.3.

1.1 Motivation

Many complex systems can be modeled as directed graphs (digraphs) or networks in applications to reveal interactions between variables, and the sparse structure is often presented in nature. An example of the latter is the interactions between species such as genes and proteins in human cells; such interactions can be modeled by stochastic/ordinary differential equations, e.g., Palsson (2011). Such network models in biology help to understand, for instance, metabolic pathways, interactions between DNAs/proteins, and furthermore contribute to pathology of, disease detection on or even clinical treatment to complicate diseases, e.g., Bar-Joseph et al. (2012). This section shows a few examples and specific concerns in data analysis that drive our studies.

1.1.1 Genetic regulatory networks

A *deoxyribonucleic acid* (DNA) molecule consists of two long polynucleotide chains composed of four types of nucleotide subunits. Each of these chains is known as a *DNA chain* or a *DNA strand*. The complete store of information in an organism's DNA is called its *gnome*, which specifies all the RNA molecules and proteins that the organism will ever synthesize. A *gene* comprises a section of DNA beginning with a *promoter* sequence and ending with a *terminator* sequence. The term *gene expression* describes the process that a cell converts the nucleotide sequence of genes first into the nucleotide sequence of RNA molecules and then into amino acid sequence of proteins, which perform functions in the cell or organisms. See Alberts (2015) for more descriptions.

The molecules, such as mRNAs and proteins, interact with each other to perform functions, which is graphically described as *genetic regulatory networks* (GRNs) in biology studies. The nodes of GRNs can represent genes, or even, in general, proteins, mRNAs, protein/protein complexes or cellular processes. Edges between nodes represent interactions between the nodes, that correspond to individual molecular reactions through which the products of one gene affect those of another. The nodes can also regulate themselves directly or indirectly via feedback loops. Figure 1.1 is an example of the GRN. To reconstruct such GRNs with gene as nodes, the measurement of nodes is in demand, which corresponds to their gene expression levels. The GRNs originate as a graphical tool to describe the information of interactions observed in biological experiments. To accelerate the reconstruction process, people associate mathematical models to GRNs, which in addition capture system behaviors and make predictions. By identifying these mathematical models from measurements, we determine the GRNs and further, if necessary, confirmed by experiments. Modeling techniques include *differential equations*, *Boolean networks*, *Bayesian networks*, *graphical Gaussian models*, etc. (Äijö and Lähdesmäki, 2009; Akutsu et al., 1999; Marbach et al., 2010; Margolin et al., 2006; Murphy and Mian, 1999; Rogers and Girolami, 2005; Song et al., 2009; Yeung et al., 2005). Each model has its strengths while having drawbacks on its descriptive

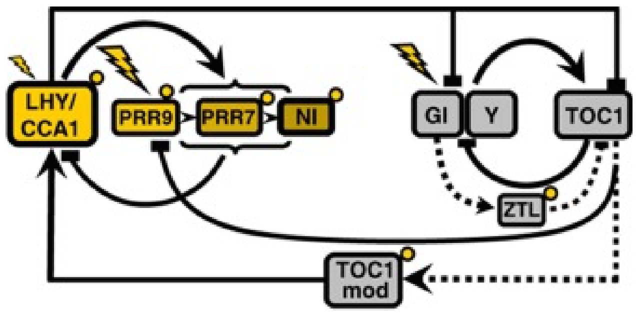


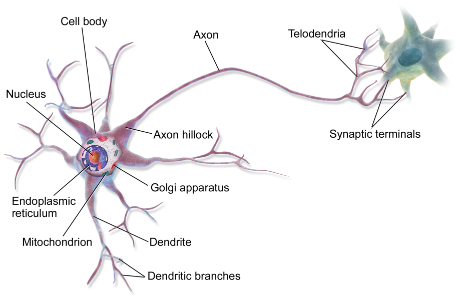
Figure 1.1. An example of genetic regulatory network, which describes genetic mechanism of circadian clock behaviors of Arabidopsis. From Pokhilko et al. (2010).

capability in certain levels, especially on delivering causal information so as to match with experimental observations. The reconstruction of GRNs becomes particularly important in biological studies, which provides interactions between genes to help understand mechanisms of biological systems, and then guides us for biological intervention/modification or treatment.

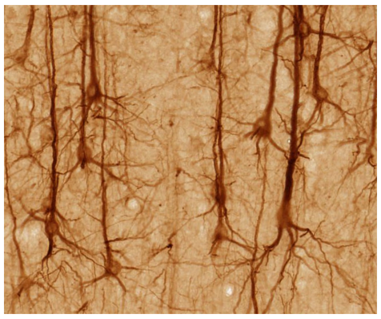
In later chapters, one specific GRN model in biology, named "Millar models" (Fogelmark and Troein, 2014; Greenham and McClung, 2015; Pokhilko et al., 2010), is going to used in numerical examples of network reconstruction. Here provides with an essential introduction of its background. The Millar models associate with a GRN that describes circadian rhythms in Arabidopsis, which is progressively explored by more than 15 years' experiments. Circadian rhythms are a biological process that displays an endogenous, entrainable oscillation of about 24 hours. The self-sustained oscillations of the clock genes drive rhythmic expression in Arabidopsis. It provides plants with an adaptive advantage to anticipate the daily changes in environmental conditions. The Arabidopsis circadian clock can be understood via a genetic network consisting of morning and evening feedback loops interconnected by a central loop (Locke et al., 2006; McClung, 2006), e.g., Figure 1.1. The specific version of Millar models, denoted *Millar-10 model*, used in our numerical simulations is provided by Pokhilko et al. (2010).

1.1.2 Nervous systems and brain networks

The nervous system is the organ system that is composed of the brain, spinal cord and nerves, and that serves as the communicating and coordinating system of the body, carrying information to the brain and relaying instructions from the brain. The basic unit of the nervous system is the nerve cell, or *neuron*, illustrated as Figure 1.2a. It is an electrically excitable cell that receives, processes, and transmits information through electrical and chemical signals. These signals between neurons occur via specialized connections called synapses. Neurons connecting to each other form neural networks, e.g., Figure 1.2b. As the most complex network known to man, a human brain comprises about 100 billion neurons



(a) neuron

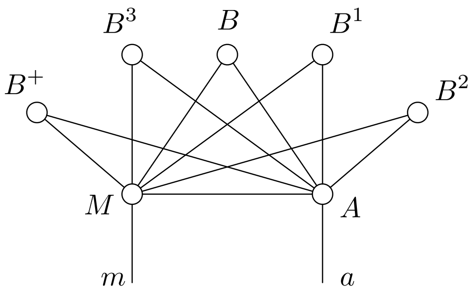


(b) neural network

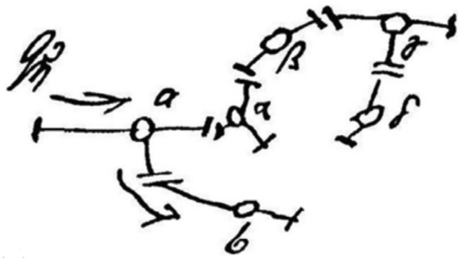
Figure 1.2. Anatomy of a multiploar neuron and SMI32-stained pyramidal neurons in cerebral cortex. **(a)** is of NeuroLex ID: `sao1417703748`, and **(b)** is from `brainmaps.org`.

connected by about 100 trillion synapses, anatomically organized over multiple scales of space and functionally interactive over multiple scales of time (Fornito et al., 2016). As a chapter title used by Buzsaki (2011), "structure defines functions", brain network connectivity has unsurprisingly been a central goal of neuroscience.

Early work of brain networks is based on clinicopathological correlations. For example, Figure 1.3a is Lichtheim's sketch of a large-scale human brain network for language, or Figure 1.3b, a diagram drawn by Freud, to map clinical idsorders onto large-scale bran networks (examples are summarized by Fornito et al. (2016)). Since the 1980s, with the



(a) Lichtheim's sketch



(b) Freud's diagram

Figure 1.3. Brain network examples based on clinicopathological correlations. **(a)**, **(b)** are reproduced or copied from Fornito et al. (2016).

development of neuroimaging techniques, including functional *magnetic resonance imaging* (MRI), *electroencephalography* (EEG), *magnetoencephalography* (MEG), *multielectrode array* (MEA) recording of local field potentials, and *positron emission tomography* (PET), people start to explore interactivity by examining neurophysiological time series. For example, (Eguiluz et al., 2005; Hagmann et al., 2008; Stam, 2004) give a graphical analyses of human brain functional networks using functional MRI and M/EEG data, in which correlation or coherence between each pair of time series is calculated and thresholded to determine an existence of edges in diagrams. An example of results by Hagmann et al. (2008) is given in Figure 1.4, where nodes are anatomical parcellations and edges represent connections. These methods allow us to access complicated structures of brain networks with less efforts

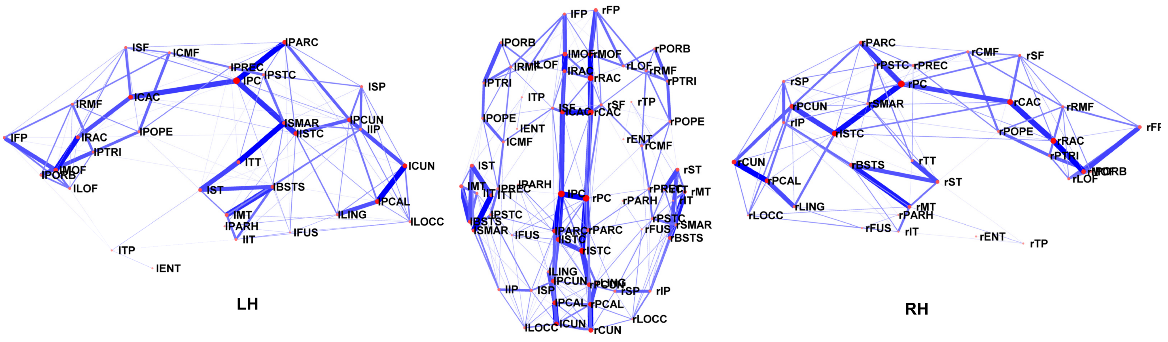


Figure 1.4. Brain networks from human MRI. From Hagmann et al. (2008).

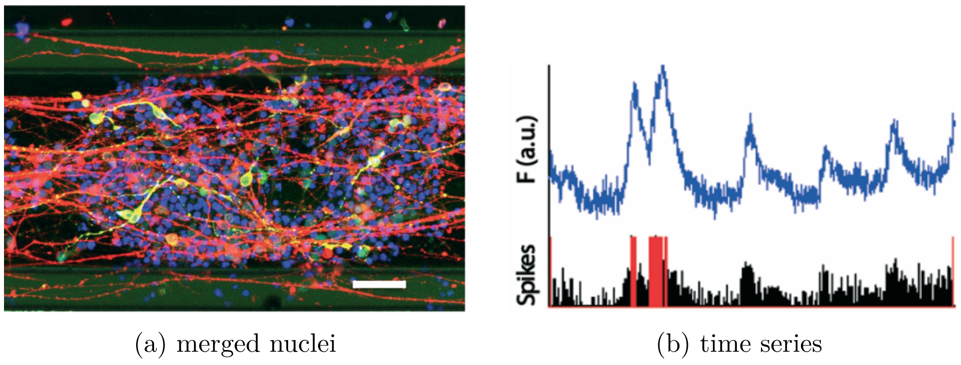
than clinicopathological correlations. However, it is questionable to conclude an connection between brain regions by statistical dependency using simple metrics like correlation. People actually have realized it and named it by *functional connectivity* to avoid this issue. (Honey et al., 2009; Skudlarski et al., 2008) have pointed out that functional connectivity is not identical to, although correlated with, anatomical connectivity. As functional connectivity is averaged over longer time periods, it may converge onto structural connectivity, although it is important to remember that structural and functional connectivity are different measures (Fornito et al., 2016; Zalesky et al., 2012). Furthermore, even in the so-called resting state, it is clear that functional connectivity is not stationary (Chang and Glover, 2010). Due to these issues, the contribution of networks of functional connectivity to cognitive function may be significantly offset in practice. Moreover, the transmission of electronic signals between neurons are directional, as implied by the mechanism of neurotransmission, which cannot be taken into account by these correlation-based networks.

As listed in the above review, these drawbacks motivate us to develop alternative network reconstruction methods, which present connectivity exactly or at least closely coinciding with anatomical connectivity and deliver more useful information to help to understand cognitive function, such as directionality, causality, orders of dynamics, delays, etc. There is no doubt that such a network reconstruction method will accelerate the understanding of neuronal systems.

1.1.3 Practical features of biological data

Time series data acquisition

Considering the dynamical nature of biological systems, time-series experiments are key to understand and model these processes. Two types of biological data are our potential applications. One is the massive calcium imaging of neurons over time using microfluidic devices, which is then converted to fluorescence time series by image processing algorithms, e.g., Figure 1.5. The other widely used time series data in biology is gene expression data.



(a) merged nuclei (b) time series

Figure 1.5. Examples of calcium imaging data and the time series. From Moreno et al. (2015).

Such data can be used to gain a wide range of insights, such as characterizing the function of specific genes, the relationships among genes, their regulation and coordination and the clinical implications of differential dynamics (Bar-Joseph et al., 2012). Most of available time-series gene expression data are generated by microarray, a multiplex lab-on-a-chip that assays large amounts of biological material using high-throughput screening miniaturized, multiplexed and parallel processing and detection methods. An example of time-series microarray data in our applications is He et al. (2012), shown in Figure 1.6. A recent improvement in bio-techniques,

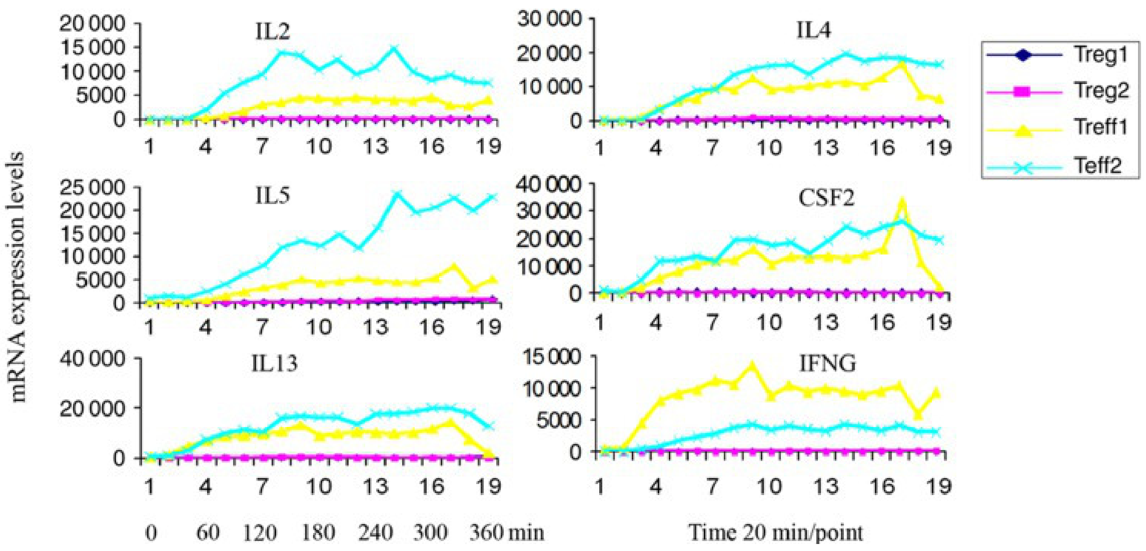


Figure 1.6. Example of microarray time series from He et al. (2012).

high-throughout RNA sequencing (RNA-seq), makes time-series expression studies more feasible and relevant. Most genomic techniques require a large quantity of biomaterials, such as populations of cells. Thus so far have time-course measurements been applied to cases in which uniform cell populations are available. Bar-Joseph et al. (2012) lists the following cases that are appropriate to assume time-series experiments: responses to external signals (such as stress (Gasch et al., 2000) or pathogens (Amit et al., 2009)), developmental processes that have a clear starting point (e.g., embryonic development (Arbeitman et al., 2002)); and cyclic internal process in which the entire cell population can be synchronized (e.g., the cell cycle (Bar-Joseph et al., 2008; Rustici et al., 2004) and circadian clock (Locke et al., 2006; McClung, 2006; Pokhilko et al., 2010)).

Sampling frequencies

Sampling rates or sampling frequencies (the reciprocal of sampling periods) are chosen depending on your goals of experiments. To capture the cyclic behaviors, e.g., circadian rhythms (Locke et al., 2006; McClung, 2006), the measurement should be sampled uniformly (called *equidistant sampling* in terminology) and cover enough number of cycles. In general, if the analysis depends on methods from time series analysis, mostly we require equidistant temporal data. Admittedly, the non-equidistant time series can also be used while in a fairly complicated way. In perturbation-response experiments most attention is given on the early stage that shows most dynamics (*transit response* in system theory), e.g., the transcriptional response in biology (Bar-Joseph et al., 2012), and sampling late time points usually do not add much information (*steady-state response* in system theory). In practice, selecting the appropriate sampling rate is difficult due to the limited prior knowledge of the systems in study. This motivates us to check, refine or re-develop methods that fail when sampling rates are not high enough.

The dynamical systems in our study are assumed to be evolving in continuous time in nature. It deserves to be emphasized that the discrete-time approach for dynamic network reconstruction (which is based on identification of dynamical systems) is valid only if the sampling frequency is high enough, where the discrete-time model shares the same network structure as the continuous one that is the physical process. To use discrete-time methods, one practical rule to choose sampling frequencies is taking ten times the bandwidth of the underlying linear systems (Ljung, 1999). However, in biological systems, most time-series data are sampled considerably slower than this empirical frequency, e.g., "high time-resolution" time series in He et al. (2012), which often cannot be solved by increasing sampling rates due to strict limits in biological experiments. These restrictions due to technology or expenses motivate us to the study in Chapter 4, 5.

Experiment repetitions

Once the general time scale is worked out, in particular in biological studies, there is the issue to make a balance between the length of time series and the number of replicates, given a fixed budget. This is also a challenging question in general, even considered for specific methods. Mostly it is "answered" by domain knowledge or experience. For example, Bar-Joseph et al. (2012) provides the following advice: "if the goal is only to identify the genes that are significantly differentially expressed at certain points in the time series, then it my be better to invest more heavily in replicates; by contrast, if the goal is to determine the complete set of genes that are changing or to investigate the pattern and kinetics of the response, a denser sampling rate is an advantage". Another example is a study on immune responses by Amit et al. (2009), which monitors a few gene expression using dense sampling over a long time period and then tries to decrease sampling rates to reach an optimal experiment design.

Provided with replicate data, the ordinary treatment is to take averages with the purpose of removing effects of noise. However, mostly, only a few replica are available in time series data sets, e.g., less than 5, which implies it no longer makes sense to work with statistical averages. Pan et al. (2015) proposes a way to take advantage of replica to increase robustness or accuracy of estimation. Yet another aspect to highlight here is that the system parameters of biological processes could be fairly different due to the variance in individuals in experiment repetitions, as observed in biological experiments He et al. (2012). The essence is the interconnection structure, e.g., interactions between genes, or communication between neurons, which are consistent over experiment repetitions. This drives us to study how to use these "heterogeneous" replicate data (system parameters could be different while their network structures keep consistent) more efficiently.

1.2 Literature Review

Network inference has been widely applied in different fields to learn interaction structures or dynamic behaviors. Such fields include systems biology, computer vision, econometrics, social networks, etc. However, there is no agreement upon the definition of "network", and it usually refers to a larger class of graphical models than the standard definition in the graph theory. In particular, with increasingly easier access to time-series data, it is expected that networks can explain dynamics or causal interactions. For instance, biologists use causal network inference to determine critical genes that are responsible for diseases in pathology.

In the study of biological Boolean network, (probabilistic) Boolean networks are introduced in (Akutsu et al., 1999; Shmulevich et al., 2002), which in essence models dynamics of the underlying processes by Markov chain models. However, this model tells little on causal relations. Due to direct inference of Boolean information, the benefit in compensation is less data for inference. Another popular model in biological network inference is Bayesian networks, e.g., Murphy and Mian (1999), whose relationships to *Hidden Markov Model* and *Boolean network* have been studied. Although it delivers certain causality information, the Bayesian network is defined on *directed acyclic graphs*. Refer to (Pearl, 1988, p. 127) for more information on domains of different probabilistic models. The feedback loops in networks could be particularly important in biological applications. Concerning general causal networks, as the primary contribution in Granger (1969), Granger causality (GC) provides causality graphs (see e.g. Eichler (2007)) based on predictability to identify causation between time-series variables. However, as it was realized by Granger (1969), this approach may be problematic in deterministic settings, especially in dynamic systems with weak to moderate coupling. Inspired by GC, which is equivalent to the *vector autoregression* form under fairly weak conditions (e.g., see Eichler (2007)), an idea of adopting system theory has been proposed to deal with causal network reconstructions. There has been several papers proposing methods for network inference by identifying simplified dynamical models in biological applications, e.g., (Beal et al., 2005; Friston et al., 2003; van Someren et al., 2000). These progresses will be reviewed in this section.

To study the identifiability issue in network inference, a general network representation for linear time-invariant (LTI) systems has been introduced by Goncalves and Warnick (2008). Similar models are proposed in different perspectives, e.g., (Van den Hof et al., 2013; Weerts et al., 2015). Results on network identifiability were firstly manifested in Goncalves and Warnick (2008). There have been many papers on the study of dynamic networks in different aspects: network identifiability (Goncalves and Warnick, 2008; Hayden et al., 2016b), network module identifiability (Van den Hof et al., 2013), network inference in discrete-time approaches (Chiuso and Pillonetto, 2012; Yue et al., 2018), etc. Our work is based on this LTI network model, called *Dynamical Structure Function* (DSF), which will be reviewed in details in Chapter 2.

1.2.1 SSM-induced graphical models

This section reviews a few graphical models based state-space models (SSMs) for network reconstruction, with additional simplifications. Consider the following LTI state-space systems in continuous time or discrete time:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t)\\ y(t) &= Cx(t) + Du(t)\end{aligned} \quad (t \in \mathbb{R}_+) \quad \text{or} \quad \begin{aligned}x(t+1) &= Ax(t) + Bu(t)\\ y(t) &= Cx(t) + Du(t)\end{aligned} \quad (t \in \mathbb{N}_+) \tag{1.1}$$

with state vector $x(t) \in \mathbb{R}^n$, input vector $u(t) \in \mathbb{R}^m$ and output vector $y(t) \in \mathbb{R}^p$. Provided with (1.1), path diagrams can be defined considering the Boolean values of system matrices. In precise, let $\mathcal{G}_{SS} = (V_{SS}, E_{SS})$ denote the digraph with vertex set

$$V_{SS} = \{x_i\}_{i=1}^{n} \cup \{u_j\}_{j=1}^{m} \cup \{y_k\}_{k=1}^{p} \tag{1.2}$$

and the directed edge set determined by the adjacency matrix

$$M_{SS} = \begin{bmatrix} A^T & 0 & C^T \\ B^T & 0 & D^T \\ 0 & 0 & 0 \end{bmatrix}, \tag{1.3}$$

where the rows (or columns) correspond to the sequence $(x_1, \ldots, x_n, u_1, \ldots, u_m, y_1, \ldots, y_p)$, respectively. Moreover, considering LTI state-space models in discrete-time, the *time series chain graph* can be defined, $\mathcal{G}_{TS} = (V_{TS}, E_{TS})$, where $V_{TS} = V_{SS} \times \mathbb{N}_+$ and

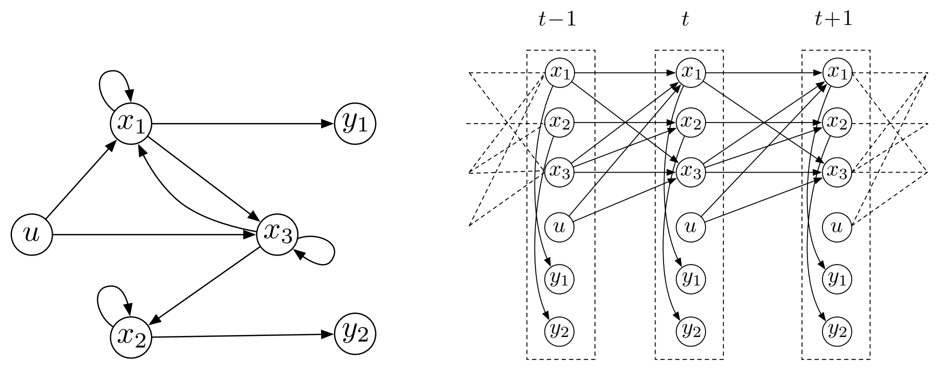
$$\begin{aligned}
\big((x_i, t-1), (x_j, t)\big) &\in E_{TS} \quad && \text{if } A_{i,j} \neq 0; \\
\big((u_i, t-1), (x_j, t)\big) &\in E_{TS} \quad && \text{if } B_{i,j} \neq 0; \\
\big((x_i, t), (y_j, t)\big) &\in E_{TS} \quad && \text{if } C_{i,j} \neq 0; \\
\big((u_i, t), (y_j, t)\big) &\in E_{TS} \quad && \text{if } D_{i,j} \neq 0.
\end{aligned}$$

There are literature on causality graphs (or causal networks) that distinguish between the following two cases: $\big((v, t-1), (z, t)\big)$ and $\big((v, t), (z, t)\big)$. They call the latter one as *contemporaneous correlation*, which is represented by an undirected edge; and hence the resultant graph becomes mixed graphs.

Example 1.1. Consider the following example

$$A = \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \\ -1 & 0 & -1 \end{bmatrix}, \; B = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \; C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \; D = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

whose digraph and time series chain graph are given in Figure 1.7.



(a) digraph (b) time series chain graph

Figure 1.7. The digraph and time series chain graph for the state-space model in Example 1.1.

Common simplifications of (1.1) include $C = I$ (states directly observed), $B = I$ (direct access to state), $D = 0$ (strictly causal interactions) and assumption on A such as symmetry (undirected graph) and sparsity (sparse graph). With full-state measurements (i.e. all state variables can be measured, which turns to be $C = I$), the above digraphs from state-space models can be directly used to reconstruct the causal interactions between measured variables, which is widely used in biological applications, as reviewed later. When the state variables are not fully measured, it raises the question on the model to describe the interactions between measured variables, which we may call *the problem of network definition*. This section reviews one reduced graphical model from state-space model, and comments on its drawbacks if considering it as a general model for network reconstructions.

Fully-observed or steady-state SSMs

The simplest networked system is the LTI state-space model under full state measurement, which is widely used in biological applications. Gardner et al. (2003) uses the state-space model with $C = I, B = I, D = 0$ to model regulatory interactions in a nin-gene subnetwork of the SOS pathway in *Escherichia coli* and correctly identifies the major genes and transcriptional targets of mitomycin C activity. Regarding network inference, it considers the steady-state case, i.e. $0 = Ax(t) + u(t)$, from which the matrix $A \in \mathbb{R}^{n\times n}$ could be obtained by solving a system linear equations given a sufficient number of independent inputs. The same steady-state model is used in Tegner et al. (2003), which presents how the perturbation of chosen genes in a microarray experiment (i.e. input design) can be used to perform a reverse engineering algorithm and reveal the underlying genetic regulatory network. There are more similar works in applications, such as (Kholodenko et al., 2002; Sontag, 2008).

Friston et al. (2003) uses a bilinear form of state-space models to model causal interactions among neuronal populations using neuroimaging time series. It starts with the fully observed nonlinear state-space model $\dot{x} = F(x, u)$, where F is some nonlinear function describing the neuro-physiological influences and u the external input. It argues that the bilinear approximation provides a natural and useful reparameterization to model the effective neuronal connectivity, given as

$$\begin{aligned} \dot{x} &\approx (A + \sum u_j L^j)x + Bu \\ A &= \frac{\partial F}{\partial x},\ L^j = \frac{\partial^2 F}{\partial x \partial u_j},\ B = \frac{\partial F}{\partial u}. \end{aligned}$$

It manifests that the Jacobian matrix A represents the first-order connectivity among the regions in the absence of input, named as “effective connectivity” or “latent connectivity”. The connectivity introduced by the parameters of the bilinear term, L^j, is called “induced connectivity”. The parameter estimation is performed by Friston et al. (2003) using the EM algorithm and full-state measurements.

Graphical model derived from SSMs

Beal et al. (2005) proposes a graphical model based on state-space models (SSMs) to reconstruct genetic regulatory networks with hidden factors. This is model is mostly referred as the method of “state-space model” for network reconstruction in biological applications, e.g., (Aderhold et al., 2014; Marbach et al., 2012). However, as seen later, it is significant simplification of SSMs. It uses the input-dependent SSM and feeds back the gene expressions from the previous time time step into the input for the current time step, which is written as

$$\begin{aligned} x(t) &= Ax(t-1) + By(t-1) + w(t), \\ y(t) &= Cx(t) + Dy(t-1) + v(t), \end{aligned} \tag{1.4}$$

where $y(t)$ denotes the gene expression levels at time t, $x(t)$ the unobserved hidden factors, and $w(t), v(t)$ are process and measurement noise. The graphical model of the setup (1.4) is given in Figure 1.8. From the equation

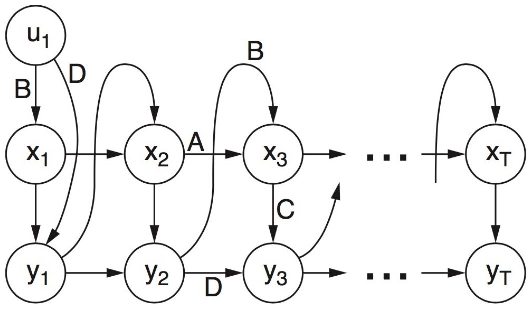


Figure 1.8. The feedback graphical model with outputs feeding into inputs, by Beal et al. (2005). Here x_k denotes the value of x at time k.

$$y(t) = (CB + D)y(t-1) + r(t),$$

where $r(t) = v(t) + Cw(t) + CAx(t-1)$, Beal et al. (2005) concentrates its analysis on the matrix $(CB + D)$, whose $(i{,}j)$-th element represents the activation or inhibition from gene j and gene i at the next time step depending on its sign. It states that the matrix $(CB + D)$ captures “all of information related to gene-gene interaction over one time step”. Considering the statistical problem of parameter estimation, it shows that if the SSM is stable, controllable and observable, then the $(CB + D)$ matrix remains invariant to any coordinate transformations of the state and is identifiable. It proposes a procedure, called *variational Bayesian EM algorithm*, to estimate parameters and hyperparameters by maximizing a lower bound of marginal likelihood.

The model for network reconstruction proposed by Beal et al. (2005) is limited in the following ways. First, the necessity to feedback outputs could be fairly difficult to implement in practice. In biological experiments, we do have control inputs available; however, they are mostly very simple signals, such as the change of light, temperature, pH, or adding toxic chemicals. Hence, supposing no questions on its thereafter network definition, the fundamental setup (1.4) covers a very special class of networked systems. Second, the focus of $(CB+D)$ to define networks is particularly restrictive. Almost all dynamics between elements of y happening via hidden states have been overlooked. The effects of hidden states/factors are not well considered in network reconstruction. Furthermore, even if we focus on the interactions between y_i over one time step, what $(CB + D)$ can capture is barely the case as Beal et al. (2005) stated ("it captures all of the information related to gene-gene interaction over one time step"). The reason is that Beal et al. (2005) ignores the information flow from y_i to y_j via hidden states, i.e. the flows enabled by A, which we believe are more important than the ones captured by $(CB + D)$. It is too restrictive that only considering the first-order dynamics between node variables in networked systems.

1.2.2 Boolean networks

The *Boolean network* (BN) model is originally introduced by (Glass and Kauffman, 1973; Kauffman, 1969), in which the gene expression is quantized to only 1 or 0 and the expression level of each gene is a function (using logical rules) of expression states of other genes. This computational model has been proven by applications, e.g., Yuh et al. (1998), to be effective in certain cases to reveal the logical rules. To cope with uncertainty of networks, attributed to increased gene complexity and differentiation-related specification, Akutsu et al. (2000) introduces noisy Boolean networks and proposes an identification algorithm, which relaxes the requirement of consistency intrinsically imposed by the Boolean functions. Furthermore, Shmulevich et al. (2002) points out that "from an empirical point of view, the assumption of only one logical rule per gene may lead to incorrect conclusions when inferring these rules from gene expression measurements, as the latter are typically noisy and the number of samples is small relative to the number of parameters to be inferred". Due this consideration, Shmulevich et al. (2002) proposes the model, *probabilistic Boolean network* (PBN), which shares the properties of Boolean networks, and is declared to be able to deal with uncertainty in both data and model selection. The essential concepts of PBN are introduced as follows.

A *Boolean network* $\mathcal{G}_{BN}(V, F)$ is defined by a set of nodes $V = \{x_1, \dots, x_n\}$ and a list of Boolean functions $F = (f_1, \dots, f_n)$. A variable x_i is called *fictitious* if

$$f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) = f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$$

for all $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$. And a variable is *essential* if it is not fictitious. The list of Boolean functions F represents the rules of regulatory interactions between genes. Given a Boolean network $\mathcal{G}_{BN}(V, F)$ and an initial joint probability distribution $D^0(x)$, $x \in \{0,1\}^n$ over the n-dimensional hypercube, the joint probability of every node after one step of the network is

$$\Pr\{f_1(x) = i_1, f_2(x) = i_2, \dots, f_n(x) = i_n\} = \sum_{\substack{x \in \{0,1\}^n; \\ f_k(x) = i_k,\, k=1,\dots,n}} D^0(x), \tag{1.5}$$

where $i_k \in \{0,1\}$. The relation (1.5) can be used in iteration fashion, which defines the iterative system

$$D^{t+1} = \Psi(D^t) \tag{1.6}$$

where the mapping $\Psi : [0,1]^{2^n} \to [0,1]^{2^n}$ is implicitly defined by (1.5). Since Ψ in fact is affine, (1.6) can be written as $D^{t+1} = D^t A = D^0 A^{t+1}$, where D^0 is the initial joint distribution, and A is a $2^n \times 2^n$ binary matrix defined as

$$A_{ij} = \begin{cases} 1 & \text{if } \exists\, x \in \{0,1\}^n, C(f_1(x), \dots, f_n(x)) = j, C(x_1, \dots, x_n) = i \\ 0 & \text{otherwise} \end{cases} \tag{1.7}$$

where $C(i_1, \dots, i_n) = 1 + \sum_{j=1}^{n} 2^{n-j} \cdot i_j$ and each $i_j \in \{0,1\}$. As easy to see, the formula of D^t is the Markov chain representation, with the binary state transition matrix A (since the state transition is specified by the Boolean functions and the probability transition can be either 0 or 1). Now we can define the *probabilistic Boolean network* $\mathcal{G}_{PBN}(V, F)$ by a set of nodes $V = \{x_1, \dots, x_n\}$ and the list $F = (F_1, \dots, F_n)$, where $F_i = \{f_j^{(i)}\}_{j=1,\dots,l(i)}$ for every node x_i, each $f_j^{(i)}$ is a possible function determining the value of gene x_i and $l(i)$ is the number of possible functions for gene x_i. A realization of the PBN at a given instant of time is determined by a vector of Boolean functions. Let $\boldsymbol{f}$ be all possible realizations of the PBN. The probability that predictor $f_j^{(i)}$ is used to predict gene i $(1 \le j \le l(i))$ is equal to

$$c_j^{(i)} = \Pr\{f^{(i)} = f_j^{(i)}\} = \sum_{k:f_{k_i}}^{(i)} \Pr\{\boldsymbol{f} = \boldsymbol{f_k}\},$$

and it must satisfy $\sum_{j=1}^{l(i)} c_j^{(i)} = 1$. The dynamics of the PBN is essentially the same as the Boolean networks. However, at any given point in time, the value of each node is determined by one of the possible predictors, chosen by its corresponding probability. In regard to the inference problem of the PBN, the *coefficient of determination* (COD) , proposed in (Dougherty et al., 2000; Kim et al., 2000), is used. Shmulevich et al. (2002) also discusses the dynamics of the PBN, such as steady-state distributions, and the relationship of the PBN to Bayesian networks.

Given the above summary of the PBN (and BN), one has been able to see its weakness, which essentially inherits from the BN model. “From a conceptual point of view, it is likely that the regularity of genetic function and interaction known to exist is not due to ‘hard-wired’ logical rules, but rather to the intrinsic self-organizing stability of the dynamical system” (Shmulevich et al. (2002)). In nature the biological systems evolve as dynamical systems over time. However, it is not clear that under what conditions the interaction structures of certain dynamical systems could coincide with the Boolean structures of the Markov chain representations $D^{t+1} = D^t A$ with binary state transition matrix A, which is inferred from system measurements. Indeed there are "successful" examples using the PBN or even BN in applications, e.g., Gardner et al. (2003) and examples listed in (Feist et al., 2009; Karlebach and Shamir, 2008). In a theoretical point of view, it is not necessary that the edges of the PBN/BN have their correspondences in physical connections. Hence, in general, one has to be particularly careful to draw conclusions on interactions from inference results of the BN or PBN. In general, in order to infer physical/causal interactions, we have to resort to identification of dynamical systems, even though only Boolean structures are desired. On the other hand, however, the PBN/BN benefits from its simplified model (binary A) for the requirement of data size in reconstruction.

1.2.3 Bayesian networks

Bayesian networks

Bayesian networks (BN) refer to *directed acyclic graphs* (DAGs), used to represent causal or temporal relationships. This term emphasizes three aspects, as summarized in Pearl (2000):

> (1) the subjective nature of the input information; (2) the relative on Bayes's conditioning as the basis for updating information; and (3) the distinction between causal and evidential modes of reasoning, a distinction that underscores Thomas Bayes's paper of 1763.

The strict definition of Bayesian networks is in essence built on a clear correspondence between the topology of a DAG and the dependence relationships portrayed by it.

First we introduce a concept that describes the topology of a DAG, presented in Pearl (1988). Let X, Y and Z be three disjoint subsets of nodes in a DAG $\mathcal{G}$, then Z is said to *d-separate* X from Y, denoted by $(X \perp\!\!\!\perp Y \,|\, Z)_G$, if there is no path between a node in X and a node in Y along which the following two conditions hold: (1) every node with converging arrows is in Z or has a descent in Z; and (2) every other node is outside Z. On the other hand, we demand the concept of conditional independence in probability theory, using the definition given by Pearl (2000). Let $V = \{V_1, V_2, \dots\}$ be a finite set of random variables, $P(\cdot)$ be a joint probability function over V, and X, Y, Z stand for any three subsets of variables in V. The sets X and Y are said to be *conditionally independent* given Z if

$$P(x|y,z) = P(x|z) \quad \text{whenever} \quad P(y,z) > 0.$$

We use a shorthand notation $(X \perp\!\!\!\perp Y \,|\, Z)_P$ or simply $(X \perp\!\!\!\perp Y \,|\, Z)$ to denote the conditional independence of X and Y given Z. And unconditional independence is denoted by $(X \perp\!\!\!\perp Y \,|\, \emptyset)$. See (Pearl, 2000, p. 11) for a list of properties of the conditional independence.

Now we are able to give a definition of Bayesian network by relating the topology of DAGs to the conditional independence in probability. A DAG $\mathcal{G}$ is said to be an *I-map* of a dependency model P if every d-separation condition displayed in $\mathcal{G}$ corresponds to a conditional independence relationship in P, i.e. if for every three disjoint sets of vertices X, Y, Z we have

$$(X \perp\!\!\!\perp Y \,|\, Z)_G \implies (X \perp\!\!\!\perp Y \,|\, Z)_P.$$

A DAG is a *minimal* I-map of P if none of its arrows can be deleted without destroying its I-mapppness. Then, *Bayesian network* of a probability distribution P on a set of variables V is defined as a DAG $G = (V, E)$ if $\mathcal{G}$ is a minimal I-map of P.

The next question is the construction of Bayesian networks (or the existence problem), which actually is pretty straightforward. Consider a distribution P on n variables $X_1, \dots, X_n$, which is assumed to admit the following identity

$$P(x_1, \dots, x_n) = \prod_i P(x_i \mid pa_i), \tag{1.8}$$

where PA_i is a select group of variables X_j $(j \neq i)$. Provided with (1.8), we can represent it by digraphs, by drawing the variables in PA_i as the parents of X_i. For example, $P(x_1, x_2, x_3, x_4, x_5) = P(x_1)P(x_2|x_1)P(x_3|x_1)P(x_4|x_2, x_3)P(x_5|x_4)$ is be represented by Figure 1.9. To be strict, the construction is summarized by the concept of *Markov compatibility*

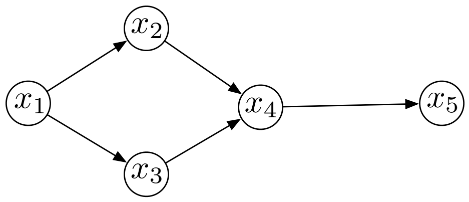


Figure 1.9. An example of DAG representing a probability function which admits a fractorization.

(see (Pearl, 2000, p. 16)). If a probability function P admits the factorization (1.8) relative to DAG $\mathcal{G}$, we say that $\mathcal{G}$ *represents* P, that $\mathcal{G}$ and P are *compatible*, or that P is *Markov* relative to $\mathcal{G}$. The set PA_i is said to be *Markovian parents* of X_j if PA_i is a minimal set of predecessors of X_j that makes X_j satisfy $P(x_i \mid x_1, \dots, x_{i-1}, x_{i+1}, x_n) = P(x_i \mid pa_i)$. The following theorem guarantees that the DAG $\mathcal{G}$ compatible to P that admits (1.8) with Markovian parents PA_i is a Bayesian network of P.

Theorem 1.1 (Pearl (2000)). *For any three disjoint subsets of nodes* (X, Y, Z) *in a DAG* $\mathcal{G}$ *and for all probability functions* P*, we have*

- $(X \perp\!\!\!\perp Y \,|\, Z)_G \implies (X \perp\!\!\!\perp Y \,|\, Z)_P$ *whenever* $\mathcal{G}$ *and* P *are compatible; and*
- *if* $(X \perp\!\!\!\perp Y \,|\, Z)_P$ *holds in all distributions compatible with* $\mathcal{G}$*, it follows that* $(X \perp\!\!\!\perp Y \,|\, Z)_G$.

The uniqueness problem studies whether two given DAGs are *observationally equivalent*, that is, whether every probability distribution that is compatible with the DAGs is also compatible with the other. It is an important property that follows from d-separation.

Theorem 1.2 (Observational Equivalence Pearl (2000)). *Two DAGs are observationally equivalent if and only if they have the same skeletons and the same sets of v-structures, that is, two converging arrows whose tails are not connected by an arrow.*

Given a joint distribution P that is Markov relative to one DAG $\mathcal{G}$, Theorem 1.2 allows us to determine which edges must be present in other DAGs that are compatible with P. Observational equivalence manifests that it is limited to infer directionality from probability alone. For example, consider the DAG in Figure 1.9. Reversing the direction of the arrow between X_1 and X_2 neither introduce nor destroy a v-structure, which hence gives an observational equivalent DAG. Without resorting to manipulative experimentation or temporal information, we cannot distinguish the new DAG from the original from probability. However, an equivalence class of network structures can be uniquely represented by a *partially directed acyclic graph* (PDAG), where a directed edge $X \to Y$ denotes that all members of the equivalence class contain the arc $X \to Y$; an undirected edge $X — Y$ denotes that the equivalence class contains either $X \to Y$ or $X \leftarrow Y$. The PDAG, representing the observational equivalence class, can be effectively constructed from the given DAG.

Learning Bayesian networks

To use Bayesian network approach in practice, the critical problem is on the learning of Bayesian networks. It consists of two levels of learning problems, as summarized in Heckerman (1998):

- given a Bayesian network, determine the probabilities of interest from complete or incomplete data (i.e. some variables are not observed);
- learning both the structure and probabilities of a Bayesian network given data.

Both of the learning problems are NP-hard, as studied by Chickering (1996). The first task is done by using Monte-Carlo methods or the Gaussian approximation to compute the posterior distribution of parameters θ for network structure G

$$P(\theta|y, G) = \sum_{z} P(z|y, G)P(\theta|y, z, G),$$

where Y and Z denote the observed and unobserved variables; and then perform the ML or MAP method. In ML or MAP estimation, one may use an approximation by ignoring the prior $P(\theta|G)$ (when the sample size increases) or find a local ML or MAP by the EM method. The second task is often more useful in practice, where the structure is mostly not unknown and is the knowledge we want to learn from data. The principle is encoding the uncertainty of network structure by defining a variable whose states correspond to the possible network structure hypotheses G. One may choose to maximize the posterior probability of a graph $P(G|D) \propto P(D|G)P(G)$ given the data D to infer the DAG (e.g., see Friedman et al. (2000)). The marginal likelihood $P(D|G) = \int P(D|G,\theta)P(\theta|G)\mathrm{d}\theta$ can be computed by Monte Carlo approach given incomplete data, e.g., (Chib, 1995; Raftery, 1996), or the Laplace approximation, e.g., Kass et al. (1988). We also need to assess the structure prior $P(G)$ and the parameter priors $P(\theta|G)$. Under certain assumptions, we can derive the structure and parameter priors for many network structures with acceptable costs, e.g., see (Buntine, 1991; Cooper and Herskovits, 1992; Heckerman and Geiger, 1995; Heckerman et al., 1995). Since finding the best G from all DAGs in which each node has no more than k parents is NP-hard for $k > 1$, we have to use heuristic search algorithms, including greedy search, best-first search, and Monte Carlo methods, etc. The key consideration for any search algorithm is the search space.

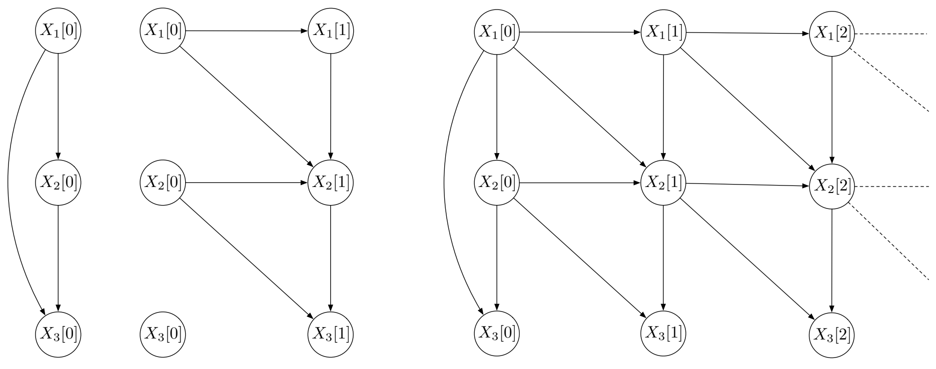
Dynamical Bayesian networks

In order to deal with temporal data and cycles in digraphs that are prevalent in practice, *dynamical Bayesian networks* (DBN), also known as *dynamic probabilistic networks* are proposed by Dean and Kanazawa (1988). Let a Bayesian network for variables X be denoted by a pair $B = (G, \theta)$, where G is the DAG and θ represents the set of parameters that quantifies the network. A Bayesian network B defines a uniqe joint probability distribution over X given by $P_B(x_1, \ldots, x_n) = \prod_i^n P_B(x_i \mid pa_i)$. The DBN is an extension of a BN over X to a family of BNs over $X[t]$, $t = 1, 2, \ldots$, which models temporal processes $X[t]$ and represents the joint distribution over all possible trajectories of a process. In most literature, e.g., (Friedman et al., 1998; Heckerman, 1998), the process is assumed to be *Markovian*, i.e., $P(X[t+1] \mid X[0], \ldots, X[t]) = P(X[t+1] \mid X[t])$, and *stationary*, i.e., the *transition probability* $P(X[t+1] \mid X[t])$ is independent of t. Given the assumptions, a DPN is defined by a pair $(B_0, B_{\rightarrow})$ that consists of two parts, referring to the definition given by Friedman et al. (1998),

- a prior network B_0 that specifies a distribution over initial state $X[0]$; and
- a transition network $B_{\rightarrow}$ over the variables $X[0], X[1]$ that is taken to specify the transition probability $P(X[t+1] \mid X[t])$ for all t.

An example is given in Figure 1.10.

Learning DBNs mirrors the results for BNs, which maximizes the marginal likelihood $P(D|G)$ to obtain the network structure and then estimate corresponding parameters. Similarly, in terms of the calculation of the marginal likelihood, there are two ways: using BIC by assuming the posteriro is insensitive to the choice of prior given a large number of data, or using a close-form solution by restricting to certain priors, e.g., BDe by using the Dirichlet distribution. The learning process demands a training data set consisting of N complete observation sequences (that requires N number of experiments). The log likelihood according to the structure of DBN is expressed as a sum of terms, each of which depends only on the



(a) DAGs of B_0 and $B_\rightarrow$ (b) DBN

Figure 1.10. An example of DBN that is specified by a prior network B_0 and a transition network $B_\rightarrow$.

conditional probability of a variable given a particular assignment to its parents (see Friedman et al. (1998) for details), and hence maximum likelihood can be done within each family independently. This decomposition property allows to learn B_0 and $B_\rightarrow$ independently, which is exactly in the same manner as learning a BN for a set of samples. The main advantage of DBNs over dynamical network is that the link dynamics are only restricted by the Markov order and therefore can be nonlinear and combinatorial. However, in practice, this advantage may be offset by the limitations of data availability. For example, the numerical example given by Yu et al. (2004) needs thousand data points to recover first-order LTI networks of 20 states. Moreover, the improvements of DBNs over BNs (i.e. dealing with cycles in digraphs and modeling temporal processes) may also be offset, considering that the learning process requires a large number of observation sequences, which implies hundreds of experiments have to be repeated and which mostly may not be viable in practice, especially in biology.

1.2.4 Granger causality

Granger causality and path diagrams

Let $X(t) = [x_1(t), ..., x_n(t)]^T, t \in \mathbb{Z}$ be a weakly stationary multivariate time series from n data channels, defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Let $V = \{1, ..., n\}$ be the set of indices. For any $A \subseteq V$, we define $X_A(t)$ as the multivariate subprocess given by the indices in A. $\bar{X}_A(t) = X_A(s), s < t$ denotes the past of the subprocess X_A at time t. Here we use the modern definition of Granger causality given by (Dahlhaus and Eichler, 2003; Eichler, 2007), which is expressed in terms of conditional orthogonality of closed linear subspaces in a Hilbert space of random variables (Eichler, 2000). See Granger (1969) for the original definition from C. W. J. Granger, which is based on comparison of variances of prediction errors.

Definition 1.3 (Granger-noncausality (Dahlhaus and Eichler, 2003)). Let A and B be disjoint subsets of V. Then

1. X_A is **Granger-noncausal** for X_B with respect to the process X_V, denoted by $X_A \nrightarrow X_B$ $[X_V]$, if
$$X_B(t) \perp \bar{X}_A(t) \mid \bar{X}_{V\setminus A}(t) \quad \forall t \in \mathbb{Z}.$$

2. X_A and X_B are **contemporaneously uncorrelated** with respect to the process X_V, denoted by $X_A \nsim X_B$ $[X_V]$, if
$$X_A(t) \perp X_B(t) \mid \bar{X}_V(t), X_{V\setminus(A\cup B)} \quad \forall t \in \mathbb{Z}.$$

Definition 1.4 (causality graph (Dahlhaus and Eichler, 2003)). The Granger Causality graph of a stationary process X is defined as a mixed graph $G = (V, E)$ that satisfies for all $a, b \in V,\ a \neq b$

1. $a \rightarrow b \notin E \Leftrightarrow X_a \nrightarrow X_b$ $[X_V]$
2. $a \!-\!\!-\! b \notin E \Leftrightarrow X_a \nsim X_b$ $[X_V]$.

The Granger Causality graphs contain two types of edges: directed edges, "$\rightarrow$", and undirected edges, "—". In the later sections we are particularly interested in "$\rightarrow$" edges.

Given the multivariate time series $X(t)$, $t \in \mathbb{Z}$, X has a moving average (MV) representation (e.g., (Brockwell and Davis, 2009; Hsiao, 1982), see (Rozanov, 1967; Skoog, 1976) for further details)
$$X(t) = \sum_{\tau=0}^{\infty} \boldsymbol{g}(\tau)\epsilon(t-\tau), \tag{1.9}$$
where $\boldsymbol{g}(\tau)$ is a square-summable sequence of $n \times n$ matrices, $\boldsymbol{g}(0) = I$; $\epsilon(t)$ is a white noise process with non-singular covariance matrix Σ. In linear systems, the sequence $\boldsymbol{g}(\tau)$ is called the *impulse response function*, given $\epsilon(t)$ as input and $X(t)$ as output. We assume that the spectral density matrix of X at frequency ω, denoted by $f(\omega)$ satisfies
$$\exists\ c \geq 1, \quad c^{-1}I \preccurlyeq f(\omega) \preccurlyeq cI \text{ for all } \omega \in [-\pi, \pi], \tag{1.10}$$
where $A \preccurlyeq B$ indicates $B - A$ is positive semidefinite. Under this assumption, the process X has a vector autoregressive (VAR) representation (see (Eichler, 2007; Rozanov, 1967), and

Masani (1966) for general conditions)

$$X(t) = \sum_{\tau=1}^{\infty} \Phi(\tau)X(t-\tau) + \epsilon(t), \tag{1.11}$$

with the non-singular covariance matrix Σ of $\epsilon(t)$.

Theorem 1.5 (Causality Graph and VAR (Eichler, 2007)). *Let X be a multivariate time series with autoregressive representation (1.11). The causality graph $G = (V, E)$ associated with X satisfies for any $a, b \in V,\ a \neq b$*

1. $a \rightarrow b \notin E \Leftrightarrow \Phi_{b,a}(\tau) = 0,\ \forall \tau \in \mathbb{N}$
2. $a \,\text{—}\, b \notin E \Leftrightarrow \Sigma_{b,a} = 0.$

There is more work devoted to generalizing the concept of causality Renault and Triacca (2015), and introducing other useful tools, e.g., *separability* (pairwise separability (Eichler, 2007; Richardson, 2003); global separability (Renault and Triacca, 2015)).

Exogeneity implications and identifiability

Considering weakly stationary $X(t)$, the existence of an econometric test for exogeneity have been demonstrated by Sims (1972), where the definition of "exogeneity" is adopted from econometric analysis. In the sense of Granger Causality, the exogenous variables U appear as those unidirectional causal for the other variables Y in X. Starting with the *complete dynamic simultaneous equation model* (CDSEM), Geweke (1978) manifests two implications of exogeneity with weaker assumptions than Sims (1972), in the equivalent concept of econometric exogeneity. Regarding Granger Causality, such a relationship could be written as a special case of the CDSEM

$$Y(t) = \Phi(q)Y(t) + \Psi(q)U(t) + \epsilon(t), \tag{1.12}$$

where $E\epsilon(t) = 0$ for all t, $\text{cov}(\epsilon(t), U(t-s)) = 0$ for all t and all $s \geq 0$, and $\text{cov}(\epsilon(t), Y(t-s)) = 0$ for all t and all $s > 0$; the vector of disturbances $\epsilon(t)$ is serially uncorrelated; the operator $\Phi(q)$ and $\Psi(q)$ are matrices of polynomials of infinite order in non-negative powers of q^{-1}, defined as *generating functions* $\Phi(q) \coloneqq \sum_{k=1}^{\infty} \Phi(k)q^{-k}$ and $\Psi(q) \coloneqq \sum_{k=0}^{\infty} \Psi(k)q^{-k}$; and the coefficients on future values of $U(t)$ are equal to zero. Note that, in the sense of Granger Causality, we only know (1.12) is valid when $Y(t)$ and $U(t)$ are jointly stationary. However, it is not necessary to require $U(t)$ to be covariance stationary when using theorems in Geweke (1978) to test exogeneity.

Theorem 1.6 (First implication of exogeneity (Geweke, 1978)). *Suppose that the regression of $Y(t)$ on all current, lagged and future values of $U(t)$ is*

$$Y(t) = K(q^{-1})X(t) + v(t) \tag{1.13}$$

where $\operatorname{cov}(v(t), U(t-s)) = 0$ *for all s and $v(t)$ is a stochastic process with autoregressive representation. Then there exists a CDSEM relating $U(t)$ and $Y(t)$ in which $U(t)$ is exogenous, if and only if, in the regression (1.13) the coefficients on future values of $U(t)$ are equal to zero.*

The study of exogeneity in (Geweke, 1978; Sims, 1972) considers the jointly stationary stochastic processes $Y(t)$ and $U(t)$. Such an exogeneity is unidirectional Granger Causality. Note that theoretical results on Granger Causality only guarantee correct identification of acyclic link structures, e.g., (Materassi and Innocenti, 2010; Materassi and Salapaka, 2012; Nabavi et al., 2015).

There is another result from Hannan (1969) for a more general representation

$$\sum_{j=0}^{m} B(j)X(t-j) = \sum_{k=0}^{n} A(k)\epsilon(t-k), \quad A(0) = I, \tag{1.14}$$

where $E\{\epsilon(j)\epsilon(k)'\} = \delta_j^k G$, the $B(j)$ and $A(k)$ are p dimensional square matrices and the generating functions

$$h(z) = \textstyle\sum_{j=0}^{m} B(j)z^j, \quad g(z) = \sum_{k=0}^{n} A(k)z^k \tag{1.15}$$

have determinants with all zeros outside of the unit circle while $h(z)$, additionally, has no zeros on the unit circle (Hannan, 1969).

Theorem 1.7 (ARMA (Hannan, 1969)). *Let* $f(\lambda) = 1/(2\pi)h(e^{i\lambda})^{-1}g(e^{i\lambda})Gg^*(e^{i\lambda})h^*(e^{i\lambda})^{-1}$. *The necessary and sufficient condition that this decomposition be unique is that a greatest common left divisor of h and g be the unit matrix and that the null space of $A(n)^T$ and $B(m)^T$ have null intersection.*

It is not trivial to see what statistical properties of data, specific network topologies, or system properties could lead to Theorem 1.7 satisfied. That is why it is difficult to use Theorem 1.7 directly, compared to sufficient conditions such as requirements on acyclic structures or sufficient perturbations in Theorem 2.14.

1.3 Preliminaries

1.3.1 System identification

System identification concerns the fundamental problem of estimation of a model from experimental data. This section is not a comprehensive review on system identification, but only the selected topics that contribute to later chapters. The contents are organized into three parts: classical treatments, a Bayesian approach, and state estimation. We use single-input-single-output (SISO) LTI systems as the example, in which $u(t)$ denotes the input that stimulates the underlying system and $y(t)$ is the output that responds to both $u(t)$ and noise/disturbance $v(t)$. System identification is modeling the underlying system and noise dynamics (usually using Gaussian white noise as its inputs), such that, for example, the prediction error $\epsilon(t)$ is small in value or is close to white noise.

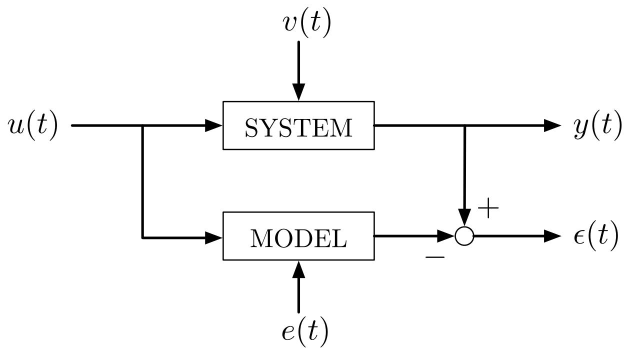


Figure 1.11. System with disturbance and diagram of system identification.

Classical system identification

Consider a LTI model in discrete time (see (Ljung, 1999, chap. 4)), given by

$$\begin{gathered} y(t) = G(q)u(t) + H(q)e(t) \\ f_e(\cdot), \text{ the PDF of } e \end{gathered} \tag{1.16}$$

with

$$G(q) = \sum_{k=1}^{\infty} g(k)q^{-k}, \quad H(q) = 1 + \sum_{k=1}^{\infty} h(k)q^{-k}, \tag{1.17}$$

where $\{g(k)\}_1^\infty$ denotes the impulse response of the underlying system, and the noise $v(t)$ is assumed to be modeled by $H(q)e(t)$. In most cases, it is usually not possible to determine all coefficients a priori, all or some of which have to be estimated, denoted by θ. The model with unknown parameters thereon is given by $y(t) = G(q,\theta)u(t) + H(q,\theta)e(t)$ with the PDF of $e(t)$, $f_e(x,\theta)$. The one-step prediction model is

$$\hat{y}(t|\theta) = H^{-1}(q,\theta)G(q,\theta)u(t) + \left[1 - H^{-1}(q,\theta)\right] y(t), \tag{1.18}$$

whose form does not depend on $f_e(x,\theta)$. We need to choose specific parametric time-series models to describe/approximate $G(q,\theta), H(q,\theta)$. For example, the autoregressive model (with extra input) models the system as

$$G(q,\theta) = \frac{B(q)}{A(q)}, \quad H(q,\theta) = \frac{1}{A(q)}, \tag{1.19}$$

where $A(q), B(q)$ are polynomials of q^{-1} with unknown coefficients to be determined, illustrated as Figure 1.12. With the specific choice of parametric models (e.g., FIR, ARX,

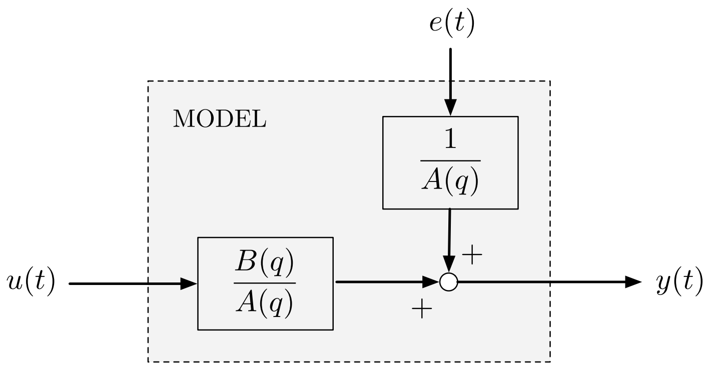


Figure 1.12. Diagram of the ARX parametric model.

ARMAX), it can be written as a (pseudo-)linear regression problem $\hat{y}(t|\theta) = \varphi^T(t,\theta)\theta$ (using $\varphi^T(t)$ when FIR or ARX), and be solved to obtain the estimation of θ. One may also formulate the parametric model in a probabilistic way and perform ML estimation. Different parametric models give different degrees of freedom to model the processes. For example, the ARX model in (1.19) obviously force to share the same roots. The selection of parametric models and the orders of parametrized models may in general be cast as *model selection* problems.

With regard to the identification of state-space models, there is the famous *subspace* method (e.g., see (Van Overschee and De Moor, 2012; Viberg, 2002)) and the gradient-based search algorithm (either PEM or ML) equipped with Kalman filters (see (Åström, 1980; Ljung and Wills, 2010)). However, considering the low sampling frequency issue, the above perspectives turn to be quite hard to be extended. The solution in Chapter 5 is built based on another space-space identification method using the EM algorithm by Gibson and Ninness (2005). It treats the state variables x as latent variables in the EM algorithm, and the complete-data log likelihood $\log p(X,Y|\theta)$ can be obtained in theory from the state space model, where X, Y denote the measurements of x, y, respectively. However, since the measurement of x is not accessible, the EM estimates the expected complete-data log likelihood $\mathbb{E}_{\theta'}[\log p(X,Y|\theta)|Y]$ provided with Y and the previous parameter estimation θ', which is maximized in each EM iteration to update the estimation of θ. To compute $\mathbb{E}_{\theta'}[\log p(X,Y|\theta)|Y]$, Kalman filters and smoothers are equipped in each EM iteration. Gibson and Ninness (2005) argues that this method does not require the smoothness of the cost function or the likelihood function, compared to most gradient-based search algorithms.

Bayesian perspectives and kernel methods

As known in classical system identification, the variance of parameter estimation increases linear with the FIR model order (Chen et al., 2012; Ljung, 1999). Hence, for higher order FIR models, it is important to counteract the increasing variance by regularization. Regularization means that we use as the cost function $\sum_t \left(y(t) - \varphi^T(t)\theta\right)^2 + \theta^T D\theta$, where the regularization matrix D is positive semi-definite. An example in MathWorks (2016) shows that even a fairly simple selection of regularization terms could help to improve the identification performance. The additional insights into the choice of D comes from Bayesian perspective, in which the parameter vector θ is treated as random variables and the regularized identification is equivalent to its MAP estimation. The regularization matrix is contributed to by prior distributions ($\theta \sim \mathcal{N}(0, P)$) and noise variance σ^2; being precise, $D = \sigma^2 P^{-1}$. It reflects the size and correlations of the impulse response coefficients (Chen et al., 2012; Pillonetto et al., 2014). Furthermore, the Bayesian perspective gives more freedom on regularization by introducing hyperprior distribution, whose parameters can be estimated by *empirical Bayes* methods. Chen et al. (2012) suggests a few useful constructions of prior covariance P in the Bayesian method:

$$\begin{aligned}
&P_{DI}(k,j) = \begin{cases} c\lambda^k & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} && \text{(diagonal)} \\
&P_{DC}(k,j) = c\rho^{|k-j|}\lambda^{(k+j)/2} && \text{(diagonal/correlated)} \\
&P_{TC}(k,j) = c\min(\lambda^j, \lambda^k) && \text{(tuned/correlated)}
\end{aligned} \tag{1.20}$$

where the hyper-parameters are $c \geq 0$, $0 \leq \lambda \leq 1$ and $|\rho| \leq 1$.

The further understanding on the construction of the regularization matrix is gained from Gaussian process (GP) applied to transfer function estimation. Pillonetto and De Nicolao (2010) applies the GP to estimate the impulse response of a stable linear system. Considering the impulse response function $\{g(k)\}_1^\infty$, the GP models it as a Gaussian process, i.e. for any finite n, $[g_1, \dots, g_n] \sim \mathcal{N}(0, P_n)$, where P_n is the $n \times n$ upper left block matrix of the semi-infinite matrix P that is defined by $\mathbf{cov}(g(t_i), g(t_j)) = P(t_i, t_j)$ ($P(t_i, t_j)$ is often called a *kernel*). With the knowledge on the standard choice of kernels in the GP, Pillonetto and

De Nicolao (2010) discusses the following kernels

$$\begin{aligned}
P_{CS}(k,j) &= \begin{cases} c(k^2/2)(j-k/3) & k \geq j \\ c(j^2/2)(k-j/3) & k < j \end{cases} && \text{(cubic spline)} \\
P_{SE}(k,j) &= c\exp(-\tfrac{(k-j)^2}{2\lambda^2}) && \text{(squared exponential)} \\
P_{SS}(k,j) &= \begin{cases} c(\lambda^{2k}/2)(\lambda^j-\lambda^k/3) & k \geq j \\ c(\lambda^{2j}/2)(\lambda^k-\lambda^j/3) & k < j \end{cases} && \text{(stable spline)}
\end{aligned} \tag{1.21}$$

where the hyper-parameters $c \geq 0$ and $0 \leq \lambda \leq 1$. There is literature, e.g., Csurcsia and Lataire (2016), which uses the perspective of regularization but constructs the regularization matrix by kernels in (1.20), (1.21), whose hyper-parameters are determined by *cross validation* or other resampling methods. Another recent work by Dinuzzo (2015) deserves our particular attention, which quantifies all kernels that stabilize the LTI systems.

State estimation: Kalman filtering and smoothing

Kalman filtering and smoothing are widely used in many fields, such as signal processing, control systems, system identification, etc. It can be presented in different perspective: least-square perspective, Bayesian perspective, etc. Here we only introduce the essential results that are going to be used in Chapter 5. The results are mainly referred to Shumway and Stoffer (2017).

Consider the linear Gaussian state-space model (for simplicity of notations, we use the shorthand x_t to denote $x(t)$)

$$\begin{aligned}
x_t &= Ax_{t-1} + w_t \\
y_t &= C_t x_t + v_t
\end{aligned} \tag{1.22}$$

where $w_t \sim \mathcal{N}(0,Q)$, $v_t \sim \mathcal{N}(0,R)$ and $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$. The primary aim is to produce estimators for the underlying unobserved signal x_t, given the data $y_{1:s} = \{y_1, \dots, y_s\}$ to time s. When $s = t$, the problem is called *filtering*; and when $s > t$, it is called *smoothing*. Let us temporarily use a shorthand notation $x_t^s = \mathbb{E}(x_t|y_{1:s})$ and $P_{t_1,t_2}^s = \mathbb{E}\left[(x_{t_1} - x_{t_1}^s)(x_{t_2} - x_{t_2}^s)^T\right]$ (when $t_1 = t_2 = t$, we use P_t^s for convenience). The *Kalman filter* is given as prediction and update steps as follows, for $t = 1, \dots, n$,

- prediction:

$$\begin{aligned}
x_t^{t-1} &= Ax_{t-1}^{t-1} + Bu_t, \\
P_t^{t-1} &= AP_{t-1}^{t-1}A^T + Q,
\end{aligned}$$

- update:

$$\begin{aligned} x_t^t &= x_t^{t-1} + K_t \left(y_t - C_t x_t^{t-1} - Du_t \right), \\ P_t^t &= (I - K_t C_t) P_t^{t-1}, \\ K_t &= P_t^{t-1} C_t^T \left(C_t P_t^{t-1} C_t^T + R \right)^{-1}. \end{aligned}$$

An important byproduct is the innovations (prediction errors) $\epsilon_t \coloneqq y_t - \mathbb{E}(y_t|y_{1:t-1}) = y_t - C_t x_t^{t-1} - Du_t$ and the corresponding covariance matrix $\mathbf{var}(\epsilon_t) = C_t P_t^{t-1} C_t^T + R$ for $t = 1, \ldots, n$. Now we consider the smoothing problem, i.e. estimating x_t based on the entire data sample $y_1, \ldots, y_n$ $(t \le n)$, which implies that each estimated value is a function of the present, future and past. The *Kalman smoother*, also known as *Rauch-Tung-Striebel smoother*, is given as, for $t = n, n-1, \ldots, 1$,

$$\begin{aligned} x_{t-1}^n &= x_{t-1}^{t-1} + J_{t-1} \left(x_t^n - x_t^{t-1} \right), \\ P_{t-1}^n &= P_{t-1}^{t-1} + J_{t-1} \left(P_t^n - P_t^{t-1} \right) J_{t-1}^T, \\ J_{t-1} &= P_{t-1}^{t-1} A^T (P_t^{t-1})^{-1}. \end{aligned}$$

In Chapter 5, we need to compute $P_{t,t-1}^n$ in the EM algorithm, which is called the *lag-one covariance smoother*, as presented below:

$$P_{t-1,t-2}^n = P_{t-1}^{t-1} J_{t-2}^T + J_{t-1} \left(P_{t,t-1}^n - A P_{t-1}^{t-1} \right) J_{t-2}^T$$

for $t = n, n-1, \ldots, 2$, with initial condition

$$P_{n,n-1}^n = (I - K_n C_n) A P_{n-1}^{n-1}.$$

1.3.2 Sparsity

The *cardinality* of $x \in \mathbb{R}^n$, denoted $\mathrm{card}(x)$, is the number of nonzero components of x. The operator card is *quasiconcave* on $\mathbb{R}_+^n$ (but not $\mathbb{R}^n$) since $\mathrm{card}(x+y) \ge \min\{\mathrm{card}(x), \mathrm{card}(y)\}$ holds for $x, y \succeq 0$, but otherwise has no convexity properties. A vector $x \in \mathbb{R}^n$ is said to be k-sparse if $\mathrm{card}(x) \le k$. A simpler notation $\|x\|_0 \le k$ is also used to denote the k-sparse x, where $\|x\|_0$ is called l_0-norm of x (strictly speaking, $\|x\|_0$ is not a *norm* of x). Informally, we may call x is sparse if $\|x\|_0 \ll n$. The nonconvexity of $\|x\|_0$ (i.e. $\mathrm{card}(x)$) raises the following two general convex-cardinality problems (with $\mathcal{C}, f$ convex)

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \|x\|_0 \\ \text{subject to} & x \in \mathcal{C} \end{array} \qquad\qquad \begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & x \in \mathcal{C}, \ \|x\|_0 \le k. \end{array}$$

(convex minimum cardinality problem) (convex problem with cardinality constraint)

Many applications in computer science and engineering can be formulated as convex-cardinality problems, for example, sparse design (finding sparest design vector x that satisfies a set of specifications), sparse modeling/regressor-selection, sparse signal reconstruction, estimation with outliers, etc. Due to the usage of basic concepts in the later chapters, we provide a bit more details on the sparse signal reconstruction problem. It considers the canonical form $y = Aw + \epsilon$, where $A \in \mathbb{R}^{N\times M}$ is a matrix whose columsn represent a possibly over-complete basis (i.e. $\text{rank}(A) = N$ and $M > N$); $w \in \mathbb{R}^M$ is the vector to be estimated; ϵ is noise, usually assumed to be $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$; and $y \in \mathbb{R}^N$ is a vector of measurements. The problem is to seek vectors whose entries are predominantly zero while still allowing us to accurately approximate y. The noise-free case is formulated as

$$\begin{array}{ll} \underset{w}{\text{minimize}} & \|w\|_0 \\ \text{subject to} & Aw = y, \end{array} \tag{1.23}$$

while the noisy case is usually solved by

$$\underset{w}{\text{minimize}}\ \|Aw - y\|_2 + \lambda\|w\|_0 \qquad \text{or} \qquad \begin{array}{ll} \underset{w}{\text{minimize}} & \|Aw - y\|_2 \\ \text{subject to} & \|w\|_0 \leq k, \end{array} \tag{1.24}$$

where $\lambda \in \mathbb{R}_+$ and $k \in \mathbb{N}$ need to be tuned by other techniques or known by prior information. However, due to $\|w\|_0$, the general convex-cardinality problem is NP-hard! It can be globally solved by *branch-and-bound*, which can work for particular problem instances with luck, but in worse case reduces to checking all (or almost all) 2^n sparsity patterns. This is barely computationally acceptable. People find more approximated methods to solve this class of problems, introduced as follows.

Convex approximation

The l_1-norm heuristic method is widely used to solve the convex-cardinality problems, studied in many fields, e.g., sparse design, LASSO and robust estimation in statistics (Friedman et al., 2010; Simon and Tibshirani, 2012; Tibshirani, 1996), support vector machine (SVM) in machine learning (Bi et al., 2003; Zhu et al., 2004), compressed sensing (Baraniuk, 2007; Donoho, 2006), total variation reconstruction in signal processing (Chambolle, 2004; Rudin et al., 1992), etc. The l_1-norm is the best convex relaxation/approximation of the l_0-norm. To see it, we first introduce the concept of *conjugate* functions. Let $f : \mathbb{R}^n \to \mathbb{R}$. The function $f^* : \mathbb{R}^n \to \mathbb{R}$, defined as

$$f^*(y) = \sup_{x\in \mathbf{dom}\, f} \left(y^T x - f(x)\right),$$

is called the *conjugate* of the function f. It is easy to see that $[f(x)]^{**} = f(x)$ if f is convex. It leads to $(\|x\|_0)^{**} = \|x\|_1$, where x is a vector in $\mathbb{R}^n$.

The next question follows naturally on the performance of l_1-norm approximation. To be precise, letting w^* be the optimal point to (1.23) and $\hat{w}$ to the following convex relaxation

$$\begin{array}{ll} \underset{w}{\text{minimize}} & \|w\|_1 \\ \text{subject to} & Aw = y, \end{array} \tag{1.25}$$

the question is to quantify the difference $\|\hat{w} - w^*\|$. It is answered by the *Restricted Isometry Property* (RIP) in Candes and Tao (2005), which characterizes matrices which are nearly orthonormal, at least when operating on sparse vectors. Let A be an $n \times m$ matrix, K an integer and a constant $\delta_K \in (0, 1)$. We say A satisfies the RIP of order K with constant δ_K if

$$(1 - \delta_K)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta_K)\|x\|_2^2 \quad \text{for all } x \text{ s.t. } \|x\|_0 \leq K.$$

The Restricted Isometric Constant (RIC) is defined as the infimum of all possible δ for a given $A \in \mathbb{R}^{n\times m}$,

$$\delta_K = \inf \left[\delta : (1 - \delta)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta)\|x\|_2^2\right] \quad \text{for any } |k| \leq K,\ \|x\|_0 \leq k.$$

Besides the general reconstruction theorem in Candes and Tao (2005), Candes and Plan (2011) provides a more convenient result, let $\delta_{2K} < \sqrt{2} - 1$, then $\|\hat{w} - w\|_2 \leq C_0\sigma(w, \|\cdot\|_1)$, where w is any feasible solution to (1.23), $C_0 \in \mathbb{R}_+$ denotes an constant, and $\sigma(w, \|\cdot\|_1)$ denotes the k-sparsity of w, i.e. $\min_z \|w - x\|_1$ s. t. $\|x\|_0 \leq k$.

Besides the standard LASSO, there are also two efficient methods based on l_2/l_1-norm heuristics, called l_2- or l_1-reweighted iterative method, by Chartrand and Yin (2008) and Candes et al. (2008), respectively. Without loss of generality, let us take l_1 reweighted method as an example (the l_2 reweight method is to replace l_1 norm of w with l_2 norm). Instead of $\min_w \|y - Aw\|_2^2 + \lambda\|w\|_1$, it

$$w^{(k+1)} \leftarrow \underset{w}{\operatorname{argmin}} \|y - Aw\|_2^2 + \lambda \sum_i \nu_i^{(k)} \|w_i\|_1, \tag{1.26}$$

where k denote the iterate index, $\nu_i^{(k)}$ denotes the weight index associated with w_i. It shows better performance than the LASSO, as claimed, analyzed and shown by examples in Candes et al. (2008).

The last remaining problem on the l_1-norm heuristics is its numerical computation, which may not be an easy problem for large-scale optimization problems and will be reviewed in Section 1.3.3.

Sparse Bayesian Learning

The general Bayesian framework is firstly proposed by Tipping (2001) to obtain sparse solution to regression and classification tasks utilizing models linear in parameters. It is thereon studied by Wipf and Rao (2004), which proves its performance on enhancing sparsity superior to l_1-heuristics. Tipping (2001) uses a zero-mean Gaussian prior distribution and utilizes Gamma distributions as hyperpriors to model the variances of the prior and posterior distributions. The uniform hyperprior, which is non-informative, is adopted in Wipf and Rao (2004); Wipf et al. (2011),which call this method *Sparse Bayesian Learning* (SBL) and whose perspective is mainly used in the later chapters. This formulation of prior distributions is a type of *automatic relevance determination* (ARD) prior (Tipping, 2001).

The SBL is to find sparse solutions of w to the classical problem $y = Aw + \epsilon$, where ϵ is noise. The SBL assumes the Gaussian likelihood, i.e. $p(y|w; \sigma^2)$, where the noise variance σ^2 is unknown and to be determined. In contrast to the previous methods, which assume a fixed prior if putting in the Bayesian framework, the SBL uses a parametric prior $p(w; \gamma)$, which depends on a vector of hyperparameters each of which controls the prior variance of each weight. For fixed values of the hyperparameters governing the prior, the posterior density of the weights $p(w|y; \gamma, \sigma^2)$ is also Gaussian. The marginal likelihood is obtained by marginalizing over the weights, and it is Gaussian and can be analytically calculated. The hyperparameters (γ, along with σ^2 if necessary) are estimated from the data by maximizing the marginal likelihood, which is referred to as *evidence maximization* or *type-II maximum likelihood*. And estimation of the weights w is obtained by performing maximum a posteriori (MAP).

To find the maximum-likelihood estimation of hyperparameters, we have to resort to numerical methods to maximize the marginal likelihood. The first method is Expectation Maximization (EM), which treats the weights w as hidden variables and then maximizing the expected complete-data likelihood $\mathbb{E}_{w|y;\gamma,\sigma^2}\left[p(y, w; \gamma, \sigma^2)\right]$, where $p(y, w; \gamma, \sigma^2) = p(y|w; \sigma^2)p(w; \gamma)$ represents the likelihood of the complete data $\{w, y\}$. We can get the closed form of E-step that is necessary to update the M-step. The expensive part is to matrix multiplication and inverse, which can be further accelerated by linear algebra modification and pruning small values of γ in EM iterates. The second method, at the expense of proven convergence, optimize the log marginal likelihood by taking the derivative with respect to γ, equating to zero, and forming a fixed-point equation that leads to faster convergence (MacKay, 1992; Tipping, 2001). The last method used in Pan et al. (2016, 2015) optimizes the cost function in the w-space using *Convex Concave Procedure* (CCCP) to obtain both w and hyperparameters, based on the essential analysis by (Wipf and Nagarajan, 2010; Wipf et al., 2011).

With regard to the performance of SBL on enhancing sparsity, the analysis in Wipf and Rao (2004) proves that the SBL is superior to both FOCUSS and l_1-norm heuristics, even though the SBL takes local minima as the solution. The work of Wipf and Nagarajan (2010) further analyzes the SBL with comparison to the l_2 reweighting method by Chartrand and Yin (2008) and the l_1 reweighting methods of Candes et al. (2008). It points out that the SBL is a non-separable reweighting scheme, in the sense that both l_2 and l_1 reweighting methods update each weight associated with each element of w separately in the weighted l_2/l_1 regularization terms in (1.24).

1.3.3 Numerical optimization

This section reviews the essential numerical methods that are going to be used in later chapters. We use the l_1-regularized least square problem as an example to review numerical methods, even though our work consider more complicated problems. In addition, it also covers the Expectation Maximization (EM), which is going to be used for evidence maximization in SBL and state-space identification in Chapter 5.

Dual problems and DCP

Let us start with a simple solution to the l_1-regularized least square problem by solving its dual problem that is standard linear-constrained quadratic program. It uses nothing more than basic convex theory in Boyd and Vandenberghe (2004). Let the primal problem be

$$\underset{w}{\text{minimize}}\ \frac{1}{2}\|Aw-y\|_2^2+\lambda\|w\|_1. \tag{1.27}$$

Introducing a dummy variable r to denote $Aw-b$, the problem (1.27) is rewritten as

$$\begin{array}{ll}\underset{w,r}{\text{minimize}} & \frac{1}{2}\|r\|_2^2+\lambda\|w\|_1\\ \text{subject to} & r=Aw-y.\end{array}$$

Introducing z as the Lagrange multipliers to write its Lagrangian, we obtain the Lagrangian dual function

$$g(z)=\inf_{w,r}\left[\|r\|_2^2/2+\lambda\|w\|_1+z^T(Aw-y-r)\right]. \tag{1.28}$$

First we simplify $g(z)$ by computing the infimum over w. Using the conjugate of the l_1-norm in (Boyd and Vandenberghe, 2004, p. 93), we obtain

$$\inf_w\left(z^TAw+\lambda\|w\|_1\right)=-\sup_w\left(-z^TAw-\lambda\|w\|_1\right)=\begin{cases}0 & \|A^Tz\|_\infty\le\lambda\\ -\infty & \text{otherwise.}\end{cases} \tag{1.29}$$

Similarly for squared l_2-norm, we have

$$\inf_r\left(\|r\|_2^2/2-z^Tr\right)=-\sup_r\left(z^Tr-\|r\|_2^2/2\right)=-\frac{1}{2}z^Tz. \tag{1.30}$$

Substituting (1.29) and (1.30) into $g(z)$, the Lagrangian dual problem can be written as a quadratic program with linear constraints

$$\begin{aligned} &\underset{z}{\text{minimize}} && \frac{1}{2} z^T z + z^T y \\ &\text{subject to} && -\lambda \leq A^T z \leq \lambda, \end{aligned} \tag{1.31}$$

which is a standard linear-constrained quadratic program and therefore can be solved by QP/QCP solvers, e.g., Gurobi, MOSEK. The optimal point x^* to (1.27) is then be obtain by $x^* = (A^T A)^{-1}(A^T y - A^T z^*)$ assuming A is column full rank, where z^* optimizes (1.31).

A more convenient choice for MATLAB users is the CVX toolbox (CVX Research, 2008; Grant and Boyd, 2015), which uses Disciplined Convex Programming (DCP) to automatically convert convex optimization problems to canonical forms and then calls standard methods, including (but not limit to) barrier, primal-dual and cutting-plane methods. One may call commercial solvers in CVX, like Gurobi and MOSEK. According to Grant (2004), the DCP deals with the *Manhattan* norm (i.e. l_1 norm) by using the following equivalence, if solving the primal,

$$\|w\|_1 \leq z \quad \Longleftrightarrow \quad \exists\, w_+, w_- \geq 0, \quad \begin{aligned} w_+ - w_- &= w \\ \mathbf{1}^T(w_+ + w_-) &= z, \end{aligned}$$

or solving the dual problem to avoid non-smoothness of the l_1-norm term in a similar manner as the example above. With the solutions to the canonical (dual) form by standard solvers, the DCP draws the *dual recovery* (see Grant (2004)) to recover solutions to the original problem. In the later chapters, the DCP is used in algorithm implementations to solve grouped reweighted l_1/l_2-regularized least-square problems, which requires tricks in CVX due to different dimensions of groups.

Proximal methods and ADMM

Proximal methods serves to solve nonsmooth, constrained, large-scale, or distributed versions of convex optimization problems. In proximal algorithms, the base operation is evaluating the *proximal operator* of a function, which involves solving a small convex optimization problem. Thereon the *proximal gradient method* and *alternative direction method of multipliers* (ADMM) can be applied to solve the whole problem. These methods will be used in Chapter 3 and 6. This section introduces the most essential definitions and algorithms, which are referred to Boyd et al. (2011); Parikh and Boyd (2013).

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a closed proper convex function. The *effective domain* of f is the set of points for which f takes on finite values, $\mathbf{dom}\, f = \{x \in \mathbb{R}^n : f(x) < +\infty\}$. The *proximal operator* $\mathbf{prox}_f : \mathbb{R}^n \to \mathbb{R}^n$ of f is defined by

$$\mathbf{prox}_f(v) = \arg\min_x \left(f(x) + (1/2)\|x - v\|_2^2 \right). \tag{1.32}$$

The function minimized on the righthand side is strongly convex and not everywhere infinite, so it has a unique minimizer for every $v \in \mathbb{R}^n$. If f is block separable, $f(x) = \sum_{i=1}^N f_i(x_i)$, then $(\mathbf{prox}_f(v))_i = \mathbf{prox}_{f_i}(v_i)$, $i = 1, \ldots, N$. Moreover, the proximal operator has the following two properties:

- (**postcomposition**) If $f(x) = \alpha\phi(x) + b$, with $\alpha > 0$, then
$$\mathbf{prox}_{\lambda f}(v) = \mathbf{prox}_{\alpha\lambda\phi}(v).$$
- (**precomposition**) If $f(x) = \phi(\alpha x + b)$, with $\alpha \neq 0$, then
$$\mathbf{prox}_{\lambda f}(v) = \frac{1}{\alpha}\left(\mathbf{prox}_{\alpha^2\lambda\phi}(\alpha v + b) - b\right).$$

In particular, if $f = \|\cdot\|_1$, then

$$(\mathbf{prox}_{\lambda f}(v))_i = \begin{cases} v_i - \lambda & v_i \geq \lambda \\ 0 & |v_i| \leq \lambda \\ v_i + \lambda & v_i \leq -\lambda \end{cases} = (1 - \lambda/|v_i|)_+ v_i, \tag{1.33}$$

called (elementwise) *soft thresholding*, where $(\cdot)_+$ replaces each negative element with 0. If $f = \|\cdot\|_2$, then $\mathbf{prox}_{\lambda f}(v) = (1 - \lambda/\|v\|_2)_+ v$, which is something called *block soft thresholding*. And if $f(x) = (1/2)\|Ax - y\|_2^2$, then $\mathbf{prox}_{\lambda f}(v) = (I + \lambda A^T A)^{-1}(v - \lambda A^T b)$. For more complicated computation of proximal operators, one may refer to Combettes and Pesquet (2011) for more algorithms, which provides the *Dykstra-like proximal algorithm* for Chapter 6.

Now let us take a look at the (accelerated) proximal gradient method and ADMM. Consider the problem

$$\underset{x}{\text{minimize}}\ f(x) + g(x), \tag{1.34}$$

where f is smooth and $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is closed proper convex. The proximal gradient method is to update x at the k-th iteration by

$$x^{k+1} := \mathbf{prox}_{\lambda^k g}(x^k - \lambda^k \Delta f(x^k)), \tag{1.35}$$

which converges with rate $O(1/k)$ when Δf is Lipschitz continuous with constant L and step size are $\lambda^k \in (0, 1/L]$. In most cases, the L may not be easy to compute, one can use the line search as presented in Algorithm 1. The *accelerated proximal gradient method* is replacing the update rule (1.35) by

$$\begin{aligned} z^{k+1} &:= x^k + w^k(x^k - x^{k-1}), \\ x^{k+1} &:= \mathbf{prox}_{\lambda^k g}(z^{k+1} - \lambda^k \Delta f(z^{k+1})), \end{aligned} \tag{1.36}$$

Algorithm 1 Line search for proximal gradient method

```
1: given $x^k, \lambda^{k-1}$ and parameter $\beta \in (0,1)$.        ▷ A typical value of $\beta$ is 1/2
2: Let $\lambda := \lambda^{k-1}$.
3: repeat
4:     Let $z := \mathbf{prox}_{\lambda g}(x^k - \lambda \Delta f(x^k))$.
5:     break if $f(z) \le \hat{f}_\lambda(z, x^k)$.    ▷ $\hat{f}_\lambda(x,y) := f(y) + \Delta f(y)^T(x-y) + (1/2\lambda)\|x-y\|_2^2$
6:     Update $\lambda := \beta\lambda$.
7: until
8: return $\lambda^k := \lambda$, $\xi^{k+1} := z$.
```

where $w^k = k/(k+3)$ and λ^k is determined similarly by line search. As a powerful algorithm suited to distributed convex optimization, the ADMM is manifested as follows

$$\begin{aligned} x^{k+1} &:= \mathbf{prox}_{\lambda f}(z^k - u^k), \\ z^{k+1} &:= \mathbf{prox}_{\lambda g}(x^{k+1} + u^k), \\ u^{k+1} &:= u^k + x^{k+1} - z^{k+1}, \end{aligned} \tag{1.37}$$

where z, u are immediate variables used in iterations.

Expectation Maximization

The *expectation maximization* (EM) algorithm is to find maximum likelihood solutions for models having latent variables. Let x be the set of all observed data, and z be the set of all latent variables. The set of all model parameter is denoted by w, and the log likelihood function is given by

$$\log p(x|w) = \log \int_z p(x, z|w) \mathrm{d}z.$$

We call $\{x, z\}$ the *complete* data set, and $\log p(x, z|w)$ the *complete-data* log likelihood function. A key observation is that the integration over the latent variables appears inside inside the logarithm, which makes the marginal distribution $p(x|w)$ fairly complicated to perform ML or MAP estimation, even if the joint distribution $p(x, z|w)$ belongs to the exponential family.

In applications of the EM algorithm, $p(x, z|w)$ is easy to compute in theory or from data if assumed to be provided with latent variables; but $p(x|w)$ is not. In practice, however, we are not given the complete data set $\{x, z\}$, but only x. To perform ML estimation on $p(x|w)$, we consider instead its expected value under the posterior distribution of the latent variable, which corresponds to the E-step of the EM algorithm. Our state of knowledge of the latent variable z is given only by the posterior distribution $p(z|x, w)$. In the E-step, we use the previous estimate w^k to update $p(z|x, w^k)$. We then use this posterior distribution to compute the expectation of the complete-data log likelihood, given by

$$\mathcal{Q}(w, w^k) = \int_z p(z|x, w^k) \log p(x, z|w) \mathrm{d}z \triangleq \mathbb{E}_{z|x,w^k} (\log p(x, z|w)) \,. \tag{1.38}$$

In the M-step, the revised estimate w^{k+1} is determined by

$$w^{k+1} = \arg\max_w \mathcal{Q}(w, w^k). \tag{1.39}$$

Each cycle of EM will increase the incomplete-data log likelihood unless it is already at a local maximum. In the MAP estimation, the E-step remains the same as in the ML case, whereas in the M-step the quantity to be maximized is given by $\mathcal{Q}(w, w^k) + \log p(w)$. Suitable choices of the prior could bring additional benefits in estimation, for example, sparse selection of parameters in Chapter 5.

1.4 Thesis Outline

1.4.1 Problem statement

The studies in this dissertation are developing methods and algorithms on dynamic network reconstruction with focus on practical issues in biological data analysis:

- limited length of time series;
- low sampling frequencies;
- heterogeneity of data sets;
- nonlinearities.

These are mostly common deficiency or particular properties of biological data, which have to be taken in account such that dynamic network reconstruction can be applied to applications. The limited size of data is considered throughout our work. Considering each of the rest properties or issues generates topics in our studies, as illustrated in Figure 1.13.

To be more specific, in the consideration of each issue, the problem is to identify the so-called *dynamical structure functions* from the limited sized of time-series data to determine Boolean networks (digraphs) or dynamic networks (digraph with capacity functions), under the assumption that the underlying network is sparse.

1.4.2 Summary of contributions

The contribution comes first is the study of fundamental methods and algorithms on dynamic network reconstructions. Even though Section 1.4.1 addresses that our work on network reconstruction is motivated by and focuses on certain biological data properties, it deserves

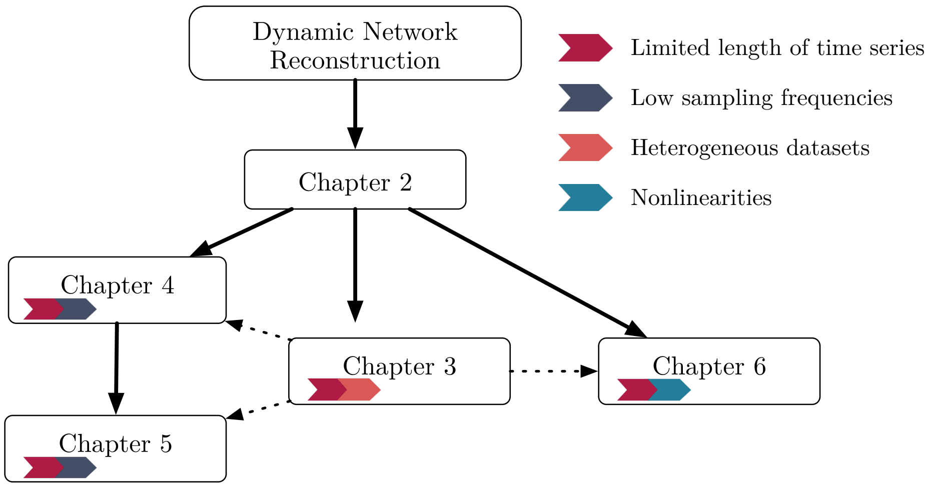


Figure 1.13. A sketch of thesis outline, where arrows represent hierarchy relations, and dash arrows implies that starting point can be applied to the end point. Each colored polygon denotes a property of biological data. The combinations of these polygons label the motivations that drive the corresponding studies.

to be stressed that the study of methods on dynamic network reconstruction (identification of the DSFs) has not yet been well studied, except the steady-state cases. And we consider the reconstruction problems in a practical scenario.

Chapter 2 starts with reviewing essential concepts on dynamical structure functions (DSF) and corresponding dynamic networks. In Chapter 2, we suggest definitions of linear network identifiability, which helps to unify the available work on network identifiability. The main results on network identifiability conditions are reviewed, as preparation for reconstruction in later chapters.

Chapter 3 addresses reconstruction of linear dynamic networks from heterogeneous datasets. Those datasets consist of measurements from linear dynamical systems in multiple experiments subjected to different experimental conditions, e.g., changes/perturbations in parameters, disturbance or noise; while the Boolean structures of the underlying networks are assumed to be same in all experiments. The network identification is performed by integrating all available datasets and promote group sparsity to assure both network sparsity and the consistency of Boolean structures over datasets. In terms of solving the problem, a treatment by the iterative reweighted l_1 method is used, together with its implementations via proximal methods and ADMM for large-dimensional networks.

Dynamic network reconstruction has been shown to be challenging due to the requirements on sparse network structures and the satisfaction of network identifiability. Moreover, the low sampling frequency makes identification problems more difficult and forces to rely on continuous-time models to infer networks. Chapter 4 raises the concept of "system aliasing", which plays a central role when the sampling frequency is not high enough. It focuses on the full-state measurement, which is supposed to the basic case but has fairly shown complications.

A sampling theorem on system aliasing is presented, together with a test criterion on system aliasing. In the case of no system aliasing, an algorithm is provided to reconstruct dynamic network from full-state measurements with large sampling periods. Moreover, it provides certain results in theory for the cases when system aliases are presented.

Considering particular difficulties on the calculation of gradients for optimization in parameter estimation in the low-sampling-frequency case, Chapter 5 uses the Expectation Minimization (EM) algorithm to bypass this issue. The E-step is calculated via Kalman filter and smoother and the M-step is equipped with Sparse Bayesian Learning (SBL) to enhance network sparsity. To solve the SBL problem, another EM algorithm is set up, in which we apply the constraints of network identifiability heuristically. Chapter 5 manages to offer a solution using Bayesian approaches to reconstruct dynamic networks from output measurements with low sampling frequency.

Systems in nature are inherently nonlinear. However, for nonlinear dynamical systems with hidden states, how to give a useful definition of dynamic networks remains an open question. Chapter 6 presents a useful definition of Boolean dynamic networks for a large class of nonlinear systems. Moreover, a robust but approximated inference method is provided. The well-known Millar-10 model in systems biology is used as a numerical example, which provides the ground truth of causal networks for key mRNAs involved in eukaryotic circadian clocks.

In a sum, this dissertation provides a class of solutions to dynamic network reconstruction considering the practical issues in biological data analysis.

Chapter 2

Dynamic Network Models

This chapter starts with essential concepts on *dynamical structure functions* (DSFs) and dynamic networks, in which we also briefly comment on other variants of network models. The suggested definitions of network identifiability is then presented, which helps to unify the available work on network identifiability. Several useful conditions on network identifiability are reviewed in Section 2.2.2, as preparation for later chapters. The last section discusses the theoretical aspects of dynamical structure functions, which includes network semantics, realizations, stability, and an graphical interpretation of paths in the definition. This chapter consists of reviews of the previous work, and extensions of DSFs and new results firstly presented here.

2.1 Linear Network Models

2.1.1 Dynamical structure function

Consider a dynamical system given by the continuous-time state-space representation in the innovations form,

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + Ke(t) \\ y(t) &= Cx(t) + Du(t) + e(t) \end{aligned} \tag{2.1}$$

where $x(t)$ and $y(t)$ are real-valued n and p-dimensional random variables, respectively; $u(t) \in \mathbb{R}^m$, A, B, C, D, K are of appropriate dimensions; and $e(t)$ is p-variate real-valued unknown white noise with covariance $\mathbb{E}[e(t)e^T(\tau)] = R\delta(t-\tau)$ and R is positive definite (denoted as $R > 0$). The initial state $x(t_0)$ is assumed to be a Gaussian random variable with unknown mean m_0 and variance R_0. The variable x is called the *state* variable, y is the *output* variable, and u is the *input* variable. Without loss of generality, we assume $n \geq p$ (i.e. output measurements constitute partial state information) and C is of full row rank. Indeed the input signals can be generated via stochastic processes in practice. The input variable

$u(t)$ is set to be a vector in $\mathbb{R}^m$, in the identification point of view[1], since we assume $u(t)$ is known or can be measured without considering measurement noise. When $C = \begin{bmatrix} I & 0 \end{bmatrix}$, the state variable x can be written as $x = \begin{bmatrix} y^T & z^T \end{bmatrix}^T$, where the variable z is called the *hidden* state variable. For convenience, we use the terms *states*, *outputs*, *inputs* and *hidden states* to refer to the elements of x, y, u and z, respectively. In later discussion on a dynamical system, we always assume the dimension of the state variable is n, which may not be known; and the dimensions of the output and input variable are p, m, respectively, which are fixed and known.

To describe the interconnections between between the variable of interests (i.e. the inputs, outputs and noise variables), the network model, known as *dynamical structure function*, is derived from (2.1) by Goncalves and Warnick (2008)

$$y(t) = Q(s)y(t) + P(s)u(t) + H(s)e(t), \tag{2.2}$$

where

$$\begin{aligned} Q(s) &= [Q_{ij}(s)]_{p\times p}, \qquad & Q_{ii}(s) &= 0, \ \forall i, \\ P(s) &= [P_{ij}(s)]_{p\times m}, \qquad & H(s) &= [H_{ij}(s)]_{p\times p}, \end{aligned}$$

where Q, P, H are $p \times p$, $p \times m$ and $p \times p$ matrices of continuous-time transfer functions, respectively; all diagonal elements of Q are zero; each element of Q (except zeros) is a strictly proper real-rational transfer function[2], and elements of P, H are proper; s is the differential operator, i.e. $sy(t) = \mathrm{d}y(t)/\mathrm{d}t$. For simplicity, we also use (Q, P, H) to denote the DSF (2.2).

Remark 2.1. We restrict the state-space models to be the innovations forms, considering the availability of definition of dynamical structure functions. It is well known that a general LTI state-space model (i.e. SSMs with process and measurement noises of different covariance matrices) can be transformed into the innovations form. However, to make a valid definition of DSFs, we have to show such a definition is invariant to the transformation.

The procedure to define the DSF (2.2) from (2.1) mainly refers to (Chetty and Warnick, 2015; Goncalves and Warnick, 2008). Without loss of generality, suppose that C is full row rank (see Chetty and Warnick (2015) for a general C). Create the $n \times n$ state transformation $T = \begin{bmatrix} C^T & E \end{bmatrix}^T$, where $E \in \mathbb{R}^{n\times(n-p)}$ is any basis of the null space of C with $T^{-1} = \begin{bmatrix} \bar{E} & E \end{bmatrix}$ and $\bar{E} = C^T(CC^T)^{-1}$. Now we change the basis such that $z = Tx$, yielding $\hat{A} = TAT^{-1}$, $\hat{B} = TB$, $\hat{C} = CT^{-1}$, $\hat{D} = D$, $K = TK$, and partitioned commensurate with the block

[1]In control theory, one may generalize $\{u(t) : t \in \mathbb{R}_+\}$ to be a stochastic process, where the input can be constructed by outputs or noise models.

[2]See Section 2.1.3 for extensions with proper Q.

partitioning of T and T^{-1} to give

$$\begin{aligned}\begin{bmatrix}\dot{z}_1(t)\\ \dot{z}_2(t)\end{bmatrix} &= \begin{bmatrix}\hat{A}_{11} & \hat{A}_{12}\\ \hat{A}_{21} & \hat{A}_{22}\end{bmatrix}\begin{bmatrix}z_1(t)\\ z_2(t)\end{bmatrix} + \begin{bmatrix}\hat{B}_1\\ \hat{B}_2\end{bmatrix}u(t) + \begin{bmatrix}\hat{K}_1\\ \hat{K}_2\end{bmatrix}e(t),\\ y(t) &= \begin{bmatrix}I & 0\end{bmatrix}\begin{bmatrix}z_1(t)\\ z_2(t)\end{bmatrix} + Du(t) + e(t).\end{aligned} \tag{2.3}$$

Introduce the differential operator s and solve for z_2, yielding

$$sz_1(t) = W(s)z_1(t) + V(s)u(t) + L(s)e(t) \tag{2.4}$$

where

$$\begin{aligned}W(s) &= \hat{A}_{11} + \hat{A}_{12}(sI - \hat{A}_{22})^{-1}\hat{A}_{21},\\ V(s) &= \hat{B}_1 + \hat{A}_{12}(sI - \hat{A}_{22})^{-1}\hat{B}_2,\\ L(s) &= \hat{H}_1 + \hat{A}_{12}(sI - \hat{A}_{22})^{-1}\hat{H}_2.\end{aligned} \tag{2.5}$$

Let $D_W(s) = \text{diag}(W(s))$ be a diagonal matrix function composed of the diagonal entries of $W(s)$ (i.e. $D_W = \text{diag}(W_{11}, W_{22}, \ldots, W_{pp})$). Define

$$\begin{aligned}\hat{Q}(s) &= (sI - D_W)^{-1}(W - D_W),\\ \hat{P}(s) &= (sI - D_W)^{-1}V,\\ \hat{H}(s) &= (sI - D_W)^{-1}L,\end{aligned} \tag{2.6}$$

yielding $z_1(t) = \hat{Q}(s)z_1(t) + \hat{P}(s)u(t) + \hat{H}(s)e(t)$. Noting that $z_1(t) = y(t) - Du(t) - e(t)$, the DSF of (2.1) with respect to y is then given by (2.2) with

$$\begin{aligned}Q(s) &= \hat{Q}(s),\\ P(s) &= \hat{P}(s) + [I - \hat{Q}(s)]D,\\ H(s) &= \hat{H}(s) + [I - \hat{Q}(s)].\end{aligned} \tag{2.7}$$

Noting that the elements of $\hat{Q}, \hat{P}, \hat{H}$ (except zeros in the diagonal of $\hat{Q}$) are all strictly proper, it is easy to see that Q is strictly proper and P, H are proper. It has been proven in Chetty and Warnick (2015) that the DSF defined by this procedure is invariant to the class of block diagonal transformations used above, which implies it is a feasible extension of the definition of DSFs given in Goncalves and Warnick (2008) for the particular class of state-space models with $C = \begin{bmatrix}I & 0\end{bmatrix}, D = 0$.

Example 2.1. Consider a system with the following transfer function

$$G = \frac{1}{s+3}\begin{bmatrix}\frac{1}{s+1}\\ \frac{1}{s+2}\end{bmatrix} \tag{2.8}$$

and its underlying state-space realization, given by

$$A = \begin{bmatrix}-1 & 0 & 1\\ 0 & -2 & 1\\ 0 & 0 & -3\end{bmatrix} \qquad \begin{aligned} B &= \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T \\ C &= \begin{bmatrix}1 & 1 & 0\end{bmatrix}. \end{aligned} \tag{2.9}$$

According to the A matrix and its corresponding DSF, we can draw the digraphs with vertices $\{y_1, y_2, x_3\}$, as shown in Figure 2.1.

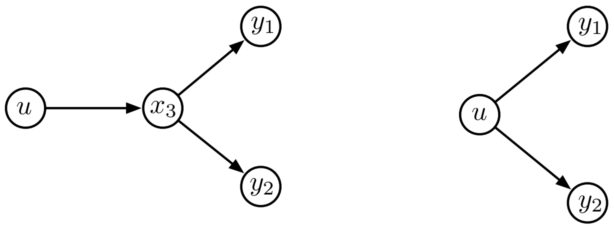


Figure 2.1. Path diagrams of Example 2.1.

Casually speaking, the digraph of the DSF (the right subfigure) is obtained from the digraph of the whole states (the left one) by removing hidden states (i.e. $\{x_3\}$) and replacing with one direct path each path from $\{u, y_1, y_2\}$ to $\{y_1, y_2\}$ via $\{x_3\}$. See Section 2.3.4 to understand how the definition of DSFs complies with the principle–removing hidden states along the information flow.

Consider the discrete-time state-space model in the innovations form

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + Ke(t)\\ y(t) &= Cx(t) + Du(t) + e(t) \end{aligned} \tag{2.10}$$

that has the same definitions of A, B, C, D, K, x_0, and $\{e(t) : t \in \mathbb{N}\}$ is a sequence of i.i.d. p-dimensional random variables with $e(t) \sim \mathcal{N}(0, R), R > 0$. The discrete-time DSF can be similarly defined from (2.10),

$$y(t) = Q(z)y(t) + P(z)u(t) + H(z)e(t) \tag{2.11}$$

where

$$\begin{aligned} Q(z) &= \hat{Q}(z), & \hat{Q}(z) &= [zI - D_W(z)]^{-1}[W(z) - D_W(z)], \\ P(z) &= \hat{P}(z) + [I - \hat{Q}(z)]D, & \hat{P}(z) &= [zI - D_W(z)]^{-1}V(z), \\ H(z) &= \hat{H}(z) + [I - \hat{Q}(z)], & \hat{H}(z) &= [sI - D_W(z)]^{-1}L(z), \end{aligned} \tag{2.12}$$

and W, V, L, D_W are defined as (2.5) with s replaced by z.

2.1.2 Dynamic networks

To make precise the path diagrams associated with the DSFs, we present the following definition by analogy with *network* in the graph theory. For convenience to give definitions for both continuous-time and discrete-time DSFs, we use q to denote either the differential operator s or the forward shift operator z.

Definition 2.1 (Yue et al. (2017)). Let $\mathcal{G} = (V, E)$ be a digraph, where the vertex set $V = \{y_1, \dots, y_p, u_1, \dots, u_m, e_1, \dots, e_p\}$[3]; the directed edge set E is defined by

- $(y_j, y_i) \in E \ \Leftrightarrow\ Q_{ij}(q) \neq 0$,
- $(u_k, y_i) \in E \ \Leftrightarrow\ P_{ik}(q) \neq 0$,
- $(e_l, y_i) \in E \ \Leftrightarrow\ H_{il}(q) \neq 0$,
- $(y_i, u_k) \notin E,\ (y_i, e_l) \notin E,\ \forall i, k, l$;

Let f be a map defined as

$$\begin{aligned} f: \quad & E \to S_{\mathrm{TF}} \\ & (y_j, y_i) \mapsto Q_{ij}(q) \ \text{ or } \ (u_k, y_i) \mapsto P_{ik}(q) \ \text{ or } \ (e_l, y_i) \mapsto H_{il}(q), \end{aligned}$$

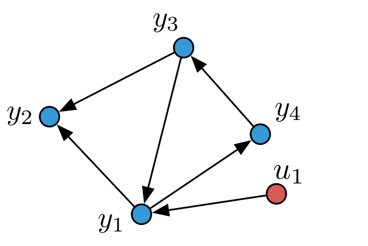
where S_{TF} is a subset of single-input-single-output (SISO) (strictly) proper rational transfer functions. We call the tuple $\mathcal{N} := (\mathcal{G}, f)$ a *linear dynamic network*, f the *capacity function* of $\mathcal{N}$, and $\mathcal{G}$ the *underlying digraph* of $\mathcal{N}$, which is also called *linear Boolean dynamic network*.

For example, the dynamic network of the deterministic DSF $y = Qy + Pu$ with

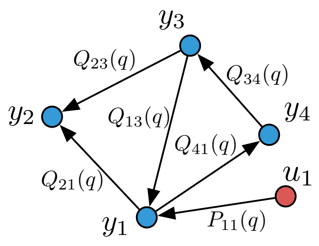
$$Q = \begin{bmatrix} 0 & 0 & Q_{13} & 0 \\ Q_{21} & 0 & Q_{23} & 0 \\ 0 & 0 & 0 & Q_{34} \\ Q_{42} & 0 & 0 & 0 \end{bmatrix}, \qquad P = \begin{bmatrix} P_{11} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.13}$$

[3]Here we use y_i, u_k, e_l as the label names of vertices, which correspond to the signals $y_i(t), u_k(t), e_l(t)$. One may alternatively define the vertex set as a set of indices.

is provided in Figure 3.2. For clarity, we also use the notation $y_j \rightarrow y_i$ to denote the directed edge (y_j, y_i) in E; similarly $u_k \rightarrow y_i$, $e_l \rightarrow y_i$. In applications the edges from u or e to y may not be interesting, one could modify the definition by removing the corresponding ones from V and E, and we still use the same terminology of "dynamic network".



(a) Boolean dynamic network



(b) dynamic network

Figure 2.2. An example of a dynamic network for a given DSF.

2.1.3 Network model variants and extensions

This section discusses a few model variants of dynamic networks that appear in the historical studies. We will discuss why these models are restrictive. Certain variants show possible ways to extend the definition of DSFs, which, however, deserve more careful considerations on their validity and meaningfulness.

Restrictive models with $P = I$

The dynamic network model proposed in Van den Hof et al. (2013) is written as

$$y(t) = Q(z)y(t) + u(t) + v(t), \tag{2.14}$$

where the non-diagonal elements of $Q(z)$ are relaxed to be proper real-rational functions, $u(t)$ is an external excitation signal that is quasi-stationary, and $v(t)$ is unmeasured disturbance being realized by a stationary stoachastic process with rational spectral density, represented by $v_j = H_j(z)e_j$ with e_j a white noise process and H_j a monic, stable and stably invertible filter. Their later works, e.g., (Weerts et al., 2016, 2015), start adopting the similar network model as the DSF, where the Q, P are relaxed to be proper. These network models in (Van den Hof et al., 2013; Weerts et al., 2016, 2015) are proposed directly, without a specific definition from the state-space models or studying the realization problem. The importance of the realization or the state-space-based definition of network models will be manifested in Chapter 5, where the reconstruction problem is so challenging that we have to transit from the state-space realizations.

It makes senses to consider the special case of H that is limited to be square, diagonal and full rank. Even though we introduce H in (2.2) as a general matrix of transfer functions, the special case of H is considered in analysis of network identifiability, e.g. Hayden et al.

(2016b). The majority of significant differences is the omission of P in the network model. The following is going to argue that this simplification leads to a fairly limited class of network models.

Without loss of generality, we consider the noise-free case, i.e. removing $v(t)$ or setting $H = 0$, and $C = \begin{bmatrix} I & 0 \end{bmatrix}, D = 0$. The model (2.14) takes the special case $P = I$, in which its state-space realization has to adopt non-zero D terms. To see this, consider the following reasoning. Let $G = \hat{G} + D$, where $\hat{G}$ is strictly proper. Consider also $\hat{P}$ to be the solution of the network from $\hat{G}$, i.e. $(I - Q)\hat{G} = \hat{P}$, and $(I - Q)G = P$. Since $P = I$ and $G = \hat{G} + D$, it yields that $(I - Q)(\hat{G} + D) = I$, or

$$(I - Q)\hat{G} + (I - Q)D = I.$$

Since $(I - Q)\hat{G} = \hat{P}$, we have $\hat{P} + (I - Q)D = I$, or

$$\hat{P} + D - QD = I.$$

From the fact that $Q, \hat{P}$ are strictly proper, if follows that $D = I$ and hence

$$\hat{P} = Q.$$

This equality implies why the model (2.14) is very restrictive. First, the assumption $P = I$ implies $\hat{P} = Q$, which tells that it would only cover a small class of LTI state-space realizations that, after setting $D = 0$, lead to $Q = P$ by definition of the DSF. Second, while inputs entering directly in the outputs y are sparse (i.e. $P = I$), there is process noise entering at all nodes where Q is nonzero (since $\hat{P} = Q$). Hence the "input" structure of disturbance/noise H is not sparse at all. Furthermore, if we considering process noises in the underlying realizations and assume Q not being diagonal, the noise part H would contradict with the setup of model (2.14), which always assumes H to be diagonal. The network could either consider the very special case Q being diagonal, which is useless, or assuming no process noise.

Extension of DSFs: general C

When we give the definition of DSFs from SSMs in Section 2.1.1, we restrict to C's of full row ranks. However, it is straightforward to consider general C's, as addressed in Chetty and Warnick (2015). Let l be the rank of C. Assume without loss of generality that the outputs $y = \begin{bmatrix} y_1^T & y_2^T \end{bmatrix}^T, y_1 \in \mathbb{R}^p$ and $y_2 \in \mathbb{R}^{(p-l)}$, are ordered such that the first l rows of C are linearly independent, $C = \begin{bmatrix} C_1^T & C_2^T \end{bmatrix}^T$ with $C_1 \in \mathbb{R}^{(l \times n)}$ being full row rank. Now we construct the state transformation by $T = \begin{bmatrix} C_1^T & E_1 \end{bmatrix}^T$, where $E_1 \in \mathbb{R}^{n \times (n-l)}$ is any basis of the null space of C_1 with $T^{-1} = \begin{bmatrix} R_1 & E_1 \end{bmatrix}$, $R_1 = C_1^T(C_1 C_1^T)^{-1}$. Performing the same state transformation as (2.3), it yields

$$\begin{bmatrix} \dot{z}_1(t) \\ \dot{z}_2(t) \end{bmatrix} = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ \hat{A}_{21} & \hat{A}_{22} \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix} + \begin{bmatrix} \hat{B}_1 \\ \hat{B}_2 \end{bmatrix} u(t) + \begin{bmatrix} \hat{K}_1 \\ \hat{K}_2 \end{bmatrix} e(t),$$
$$y(t) = \begin{bmatrix} I & 0 \\ \hat{C}_{21} & 0 \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix} + \begin{bmatrix} \hat{D}_1 \\ \hat{D}_2 \end{bmatrix} u(t) + \begin{bmatrix} \hat{I}_1 \\ \hat{I}_2 \end{bmatrix} e(t),$$

where $\begin{bmatrix} \hat{I}_1^T & \hat{I}_2^T \end{bmatrix}^T$ is the identity matrix. Following the same procedure (2.4)-(2.7) gives the DSF model $\begin{bmatrix} y_1^T(t) & y_2^T(t) \end{bmatrix}^T = Q(s)y_1(t) + P(s)u(t) + H(s)e(t)$, where

$$\begin{aligned} Q(s) &= \begin{bmatrix} \hat{Q}(s) \\ \hat{C}_{21} \end{bmatrix}, \\ P(s) &= \begin{bmatrix} \hat{P}(s) + [I - \hat{Q}(s)]\hat{D}_1 \\ \hat{D}_2 - \hat{C}_{21}\hat{D}_1 \end{bmatrix}, \\ H(s) &= \begin{bmatrix} \hat{H}(s) + [I - \hat{Q}(s)]\hat{I}_1 \\ \hat{I}_2 - \hat{C}_{21}\hat{I}_1 \end{bmatrix}, \end{aligned} \tag{2.15}$$

and $W, V, L, \hat{Q}, \hat{P}, \hat{H}$ are defined same as (2.5), (2.6). It is clearly that, due to real matrix $\hat{C}_{21}$, the $Q(s)$ is now proper (i.e. each element of Q, except zeros, is proper but may not be strictly proper), instead of strictly proper. Note that $\hat{Q}$, as the upper partition of Q, keeps being strictly proper. Moreover, Q is no longer hollow (i.e. all diagonal elements are zero). The matrices P, H are proper, which are the same as the aforementioned. The only unique property is that the $(p-l)$ rows of P, H at bottom are always real.

2.2 Network Identifiability

There have been many studies on linear dynamic network identifiability, e.g., (Gevers et al., 2016; Goncalves and Warnick, 2008; Hayden et al., 2016b), with different perspectives on this concept. However, a strict definition in mathematics is still in discussion. Here we present a definition, which could unify the available results in a common language.

2.2.1 Definitions

Consider the network model (2.2), which yields

$$\begin{aligned} y(t) &= (I - Q(q))^{-1}P(q)u(t) + (I - Q(q))^{-1}H(q)e(t) \\ &\triangleq G_u(q)u(t) + G_e(q)e(t). \end{aligned} \tag{2.16}$$

Referring to (Ljung, 1999, chap. 4), we have the one-step-ahead prediction of y, $\hat{y}(t|t-1) = G_e^{-1}G_u u(t) + (I - G_e^{-1})y(t)$. Substituting the expressions of G_u, G_e in terms of Q, P and H, it yields the one-step-ahead predictor of DSFs, named *network predictor model*,

$$\hat{y}(t|t-1) = H^{-1}(q)P(q)u(t) + H^{-1}(Q(q) + H(q) - I)y(t). \tag{2.17}$$

Given any class of parametric models, the corresponding network predictor models can be obtained by substituting in (2.17) the parametric expressions of Q, P and H.

Definition 2.2. A *network predictor model* of an LTI system is a stable filter $W(q)$ that defines a predictor as follows

$$\hat{y}(t|t-1) = W(q)r(t), \tag{2.18}$$

where

$$W(q) = \begin{bmatrix} W_u(q) & W_y(q) \end{bmatrix}, \; r(t) = \begin{bmatrix} u(t) \\ y(t) \end{bmatrix}, \tag{2.19a}$$

$$W_u(q) = H^{-1}(q)P(q), \; W_y(q) = H^{-1}\left[Q(q) + H(q) - I\right]. \tag{2.19b}$$

Consulting the definition of identifiability of input-output models in Ljung (1999), the version for linear dynamic networks is presented as follows.

Definition 2.3. A *network model set* $\mathcal{M}^*$ and *network predictor model set* $\mathcal{M}^*_{\text{Pred}}$ are defined as follows

$$\mathcal{M}^* = \left\{ (Q(q), P(q), H(q))_\iota : \iota \in \mathcal{I} \right\}, \tag{2.20a}$$

$$\mathcal{M}^*_{\text{Pred}} = \{ W_\iota(q) : \iota \in \mathcal{I} \}, \tag{2.20b}$$

where $\mathcal{M}^*_{\text{Pred}}$ is a set of $W(q)$ that is defined in (2.19), and $\mathcal{I}$ denotes the general index set, which is not necessary to be finite nor countable.

Note that $\mathcal{I}$ can be any index set, which is not necessary to be finite or countable. Here we modify the definition of model set by replacing predictor models with dynamical structure functions. In network reconstruction applications, besides model prediction, network topology also matters, which is encoded in (Q, P, H) instead of W.

Definition 2.4. A *network model structure* $\mathcal{M}$ and a *network predictor model structure* $\mathcal{M}_{\text{Pred}}$ are differentiable maps from a connected open subset $D_{\mathcal{M}} \subseteq \mathbb{R}^d$ to model sets

$$\begin{aligned} \mathcal{M}: \quad D_{\mathcal{M}} &\to \mathcal{M}^* \\ \theta &\mapsto (Q(q,\theta), P(q,\theta), H(q,\theta)), \end{aligned} \tag{2.21a}$$

$$\begin{aligned} \mathcal{M}_{\text{Pred}}: \quad D_{\mathcal{M}} &\to \mathcal{M}^*_{\text{Pred}} \\ \theta &\mapsto W(q,\theta), \end{aligned} \tag{2.21b}$$

such that the gradient of the predictor model is stable, i.e. for any given $z \in \mathbb{C}$, $|z| \geq 1$, the gradient of $W(z,\theta)$ over θ exists and is stable for $\theta \in D_{\mathcal{M}}$, where

$$W(z,\theta) = \begin{bmatrix} H^{-1}(z,\theta)P(z,\theta) & H^{-1}(z,\theta)[Q(z,\theta) + H(z,\theta) - I] \end{bmatrix}.$$

Now we present the definitions of network identifiability.

Definition 2.5 (network identifiability). A network model structure $\mathcal{M}$ is *globally identifiable at* θ^* ($\theta^* \in D_{\mathcal{M}}$) if

$$\mathcal{M}_{\text{Pred}}(\theta) = \mathcal{M}_{\text{Pred}}(\theta^*),\ \theta \in D_{\mathcal{M}} \ \Rightarrow\ \theta = \theta^*. \tag{2.22}$$

Moreover, $\mathcal{M}$ is *globally identifiable* if it is globally identifiable at almost all $\theta^* \in D_{\mathcal{M}}$, i.e. the set of θ at which $\mathcal{M}$ is not globally identifiable has Lebesgue measure zero.

The local version of network identifiability can be defined similarly as (Ljung, 1999, p. 113). We say $\mathcal{M}$ is *locally identifiable at* θ^* if there exists $\delta > 0$ such that

$$\mathcal{M}_{\text{Pred}}(\theta) = \mathcal{M}_{\text{Pred}}(\theta^*),\ \theta \in N_\delta(\theta^*) \ \Rightarrow\ \theta = \theta^*, \tag{2.23}$$

where $N_\delta(\theta^*)$ denotes a neighborhood of θ^* with the radius δ. In addition, $\mathcal{M}$ is *locally identifiable* if it is locally identifiable at almost all $\theta^* \in D_{\mathcal{M}}$.

To put other available perspectives on network identifiability in a unified language (Gevers et al., 2016; Goncalves and Warnick, 2008), we introduce two more definitions by modifying (2.22) in Definition 2.5.

Definition 2.6 (network identifiability with known structures). A network model structure $\mathcal{M}$ is *globally identifiable at* θ^* *with known structure* ($\theta^* \in D_{\mathcal{M}}$) if

$$\mathcal{M}(\theta) = \mathcal{M}(\theta^*),\ \theta \in D_{\mathcal{M}} \ \Rightarrow\ \theta = \theta^*; \tag{2.24}$$

Moreover, $\mathcal{M}$ is *globally identifiable with known structure* if it is globally identifiable at almost all $\theta^* \in D_{\mathcal{M}}$, i.e. the set of θ at which $\mathcal{M}$ is not globally identifiable has Lebesgue measure zero.

Definition 2.7 (alternative network identifiability: unique factorization). A network model structure $\mathcal{M}$ is *globally identifiable at* θ^* ($\theta^* \in D_{\mathcal{M}}$) if

$$\mathcal{M}_{\text{Pred}}(\theta) = \mathcal{M}_{\text{Pred}}(\theta^*),\ \theta \in D_{\mathcal{M}} \ \Rightarrow\ \mathcal{N}(\theta) = \mathcal{N}(\theta^*), \tag{2.25}$$

where $\mathcal{N}(\theta)$ denotes such a $\mathcal{N}$ that is determined from $[Q(\theta), P(\theta), H(\theta)]$ by Definition 2.1.

Moreover, $\mathcal{M}$ is *globally identifiable* if it is globally identifiable at almost all $\theta^* \in D_{\mathcal{M}}$, i.e. the set of θ at which $\mathcal{M}$ is not globally identifiable has Lebesgue measure zero.

Remark 2.2. Note that $\mathcal{N}(\theta) = \mathcal{N}(\theta^*)$ is equivalent to $\mathcal{M}(\theta) = \mathcal{M}(\theta^*)$. This implies (2.24) and (2.25) together are Definition 2.5.

Compared to the definition of input-output system identifiability (Ljung, 1999, p. 112,114), one may notice that Definition 2.6 is obtained by replacing the transfer function model set[4] in (Ljung, 1999, p. 114) by the network model set. Definition 2.6 can be interpreted as a definition of networks identifiability with known structures (i.e. the Boolean networks are known). The difference between (2.24) and (2.22) has to be emphasized. The model structures $\mathcal{M}_{\text{Pred}}$ and $\mathcal{M}$ are not equivalent. It is due to the fact that we need additional conditions to guarantee the unique factorization of $(Q(q,\theta), P(q,\theta), H(q,\theta))$ from $W(q,\theta)$, which is mathematically described by Definition 2.7. This is the essential challenge in network reconstruction, which has been used as a definition of "network identifiability" in (Gevers et al., 2016; Goncalves and Warnick, 2008; Hayden et al., 2016b). It is "reasonable" in the sense that in most applications we are particularly interested in the conditions to guarantee the unique identification of $\mathcal{N}$ from data (even if the exactly same $\mathcal{N}$ can be determined by different parameters θ's).

Considering *Boolean dynamic networks*, it is likely that multiple θ's lead to the same Boolean structure. However, the essence is to guarantee unique Boolean structures from

[4]Here we consider the model set as a family of $(G(q), H(q))$ instead of $W(q)$ in (Ljung, 1999, p. 107), by noticing that they are actually equivalent for input-output models (Ljung, 1999, p. 105).

data. Therefore, the definition follows by replacing the dynamic network $\mathcal{N}$ in (2.25) with its digraph $\mathcal{G}$.

Definition 2.8 (Boolean network identifiability). The Boolean structure $\mathcal{G}$ of a dynamic network with model structure $\mathcal{M}$ is globally identifiable at θ^* ($\theta^* \in D_{\mathcal{M}}$) if

$$\mathcal{M}_{\text{Pred}}(\theta) = \mathcal{M}_{\text{Pred}}(\theta^*),\ \theta \in D_{\mathcal{M}} \ \Rightarrow\ \mathcal{G}(\theta) = \mathcal{G}(\theta^*). \tag{2.26}$$

Moreover, the Boolean network is globally identifiable if it is globally identifiable at almost all $\theta^* \in D_{\mathcal{M}}$.

Example 2.2. Let us continue with Example 2.1 to show the necessity of identifiability study. Provided with $G(s)$ in (2.8), the input-output behavior has been determined. However, due to the redundancy of realization, besides the network structure in Figure 2.1 corresponding to A in (2.9), there is an alternative state-space realization

$$A = \begin{bmatrix} -2 & -1 & 1 \\ -1 & -3 & 1 \\ 0 & -1 & -1 \end{bmatrix} \qquad \begin{aligned} B &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \\ C &= \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}. \end{aligned} \tag{2.27}$$

and network structure as Figure 2.3.

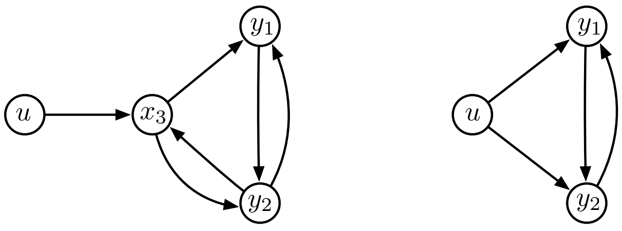


Figure 2.3. Path diagrams of Example 2.2.

Obviously it is neither network identifiable nor Boolean network identifiable. We are unable to reconstruct the ground truth no matter how much output measurement is acquired with the current input setup (by which we mean $u(t)$ only perturbs x_3).

2.2.2 Identifiability conditions

This section reviews the network identifiability studies by (Goncalves and Warnick, 2008; Hayden et al., 2016a,b), which provides the essential identifiability conditions in the future reconstruction methods. These studies use Definition 2.7 and choose the model set $\mathcal{M}^*$ to be a set of all (Q, P, H) whose dimensions are determined by the fixed dimensions of output and input variables and each element is a strictly proper real-rational transfer function (of arbitrary orders). The original studies are manifested in the frequency domain, whereas we rephrase them in the time domain for convenience in later chapters. And all the studies reviewed in this section consider the DSFs derived from the SSMs with $C = \begin{bmatrix} I & 0 \end{bmatrix}, D = 0$ and no measurement noise.

Unique factorization from transfer functions

The work in Goncalves and Warnick (2008) studies the "deterministic" case (Q, P), whereas the stochastic case can also be applied by replacing the inputs $u(t)$ with stochastic processes, as studied in Hayden et al. (2016b). The major results on network identifiability in Goncalves and Warnick (2008) are presented as the following theorem and corollary.

Theorem 2.9 (Reconstruction with Partial Structure Goncalves and Warnick (2008)). *Given a $p \times m$ transfer function G, dynamical structure reconstruction is possible from partial structure information if and only if $p-1$ elements in each column of $\begin{bmatrix} Q & P \end{bmatrix}^T$ are known that uniquely specify the component of (Q, P) in the nullspace of $\begin{bmatrix} G^T & I \end{bmatrix}$.*

Corollary 2.10 (Goncalves and Warnick (2008)). *If $m = p$, G is full rank, and there is no information about the internal structure of the system Q, then the dynamical structure can be reconstructed if each input controls a measured state independently (i.e., without loss of generality, the inputs can be numbered such that P is diagonal). Moreover, $L = G^{-1}$ characterizes the dynamical structure as follows:*

$$Q_{ij} = -\frac{L_{ij}}{L_{ii}} \text{ and } P_{ii} = \frac{1}{L_{ii}}.$$

These results can be extended to the general case (Q, P, H) without difficulty if the transfer matrices G_u, G_e, defined in (2.16) are assumed to be known. For instance, Corollary 2.10 can be straightforwardly extended to Corollary 2.11 for (Q, P, H). Note that this assumption might not be trivially satisified in practical identification, where we may only be able to access $G_e(q)G_e^{-*}(q)$ instead of $G_e(q)$, which raises additional issues on network identifiability and has been studied in Hayden et al. (2016b).

Corollary 2.11 (Extension of Corollary 2.10 for (Q, P, H)). *Given $p \times m$ and $p \times p$ transfer matrices G_u, G_e, the dynamical structure (Q, P, H) can be reconstructed if one of the following conditions holds:*

- *$m = p$, G_u is full rank, and P is diagonal and full rank*[5]*;*
- *G_e is full rank, and H is diagonal and full rank.*

[5]This is a simplified description of the condition "each input controls a measured state independently" in Corollary 2.10 without loss of generality.

Moreover, if G_u is full rank and P is diagonal and full rank, it yields

$$Q_{ij} = -\frac{L_{ij}}{L_{ii}},\ P_{ii} = \frac{1}{L_{ii}},\ \textit{and}\ H = (I - Q)G_e,$$

where $L = G_u^{-1}$; whereas if G_e is full rank and H is diagonal and full rank, the dynamical structure follows similarly by setting $L = G_e^{-1}$.

To comment on the strong conditions on network identifiability from Goncalves and Warnick (2008), we need to introduce a concept from system identification (see (Ljung, 1999, p. 110)) and modify it for network reconstruction.

Definition 2.12. A network model structure $\mathcal{M}$ is said to be *independently parametrized* if

$$\mathcal{M}(q,\theta) = (Q(q,\xi), P(q,\eta), H(q,\zeta)),\ \theta = \left[\xi^T\ \eta^T\ \zeta^T\right]^T,$$

where $D_{\mathcal{M}} = D_\xi \times D_\eta \times D_\zeta$, $\theta \in D_{\mathcal{M}}$, $\xi \in D_\xi$, $\eta \in D_\eta$ and $\zeta \in D_\zeta$.

In most practice, we may not use the independently parametrized model structures. For example, the ARX model (denoted as $\mathcal{M}^{\text{ARX}}$) and the ARMAX model ($\mathcal{M}^{\text{ARMAX}}$) are not independent parametrizations (see Yue et al. (2018) for the details on both models applied for linear dynamic networks). Actually $\mathcal{M}^{\text{ARX}}$ is globally identifiable by noticing that the parameter estimation via *prediction error minimization* (PEM) is equivalent to solve a *quadratic program* (QP), as long as ground truth is in this model set.

Consider the work in Goncalves and Warnick (2008), which studies the most general case of independent parametrization. It studies the network model set $\mathcal{M}$ that contains all linear network models $(Q(s), P(s))$[6] for LTI systems, and each element (i.e. a SISO strictly proper transfer function) is parametrized independently. Let θ denote the parameter vector of $\mathcal{M}$, which might be of an infinite dimension. The work in Goncalves and Warnick (2008) studies what conditions guarantee that $(Q(s), P(s))$ can be uniquely factorized from $G(s)$, which seems only discussing Definition 2.7 instead of Definition 2.5. However, the philosophy in Goncalves and Warnick (2008) is that: assuming that the model structure of $G(s, \theta_{\text{TF}})$ (θ_{TF} denotes the parameterization of transfer functions; the range of the model structure is the set of all linear transfer functions) is identifiable, if $(Q(s,\theta), P(s,\theta))$ can be uniquely determined from $G(s, \theta_{\text{TF}})$, then the unique solution to θ follows from θ_{TF}. This perspective avoids repeating the identifiability issues of transfer functions that have been soundly studied.

The results in Goncalves and Warnick (2008) addresses the conditions under which $(Q(q), P(q))$ can be uniquely factorized from $G(q)$. Let us understand in the following way: assuming the model structure of $G(q, \theta_{\text{TF}})$ (θ_{TF} denotes the parameterization of transfer

[6]The study in Goncalves and Warnick (2008) is performed in frequency domain.

functions) is identifiable, then $(Q(q,\theta), P(q,\theta))$ is uniquely determined from $G(q,\theta_{\text{TF}})$ directly, and a unique solution to θ follows from θ_{TF}. Therefore, such conditions guarantee both Definition 2.7 and 2.6 satisfied. The work in Goncalves and Warnick (2008), i.e. studying the uniqueness of determining $(Q(q), P(q))$ from $G(q)$ without parameterization, can be understood as guaranteeing Definition 2.7 under the general parameterization θ.

Unique factorization from spectral density

The work in Hayden et al. (2016b) studies "network identifiability" from intrinsic noises. To be precises, it studies the conditions to guarantee the unique factorization of (Q, H) from the spectral density of outputs[7]. It considers the standard LTI state-space representation (A, B, C, D), where the input $u(t)$ is white noise; and uses $G(s)$ to refer to the transfer function from $u(t)$ to the output $y(t)$. The output spectral density is denoted by $\Phi(s) = G(s)G^{-*}(s)$. The major theorem on network identifiability depends on a family of assumptions, which are summarized as follows.

Assumption 2.1. The matrix A is *Hurwitz*.

Assumption 2.2. The system is driven by unknown white noise $u(t)$ with covariance $\mathbb{E}[u(t)u^T(\tau)] = I\delta(t-\tau)$.

Before giving Assumption 2.3, we need a new concept from Hayden et al. (2016b).

Definition 2.13 (Global Minimality (Hayden et al., 2016b)). For a given spectral density $\Phi(s)$, the *globally minimal degree* is the smallest degree of all its spectral factors. Any system (or realization) of globally minimal degree is said to be *global minimal*.

Assumption 2.3. The system (A, B, C, D) is globally minimal.

Assumption 2.4. The matrices $C = [I \;\; 0]$ and $D = 0$.

Assumption 2.5. The matrix H is square, diagonal and full rank.

Assumption 2.6. The transfer function $G(s)$ is minimum phase.

Under the above setup, Hayden et al. (2016b) presents an important identifiability conclusion, which essentially tells whether (Q, H) could be uniquely factorized, up to signs of $H(s)$, from the output spectral density.

[7] Hayden et al. (2016b) originally uses the notation (Q, P), where the inputs have been selected to be white noise. In our general notation (Q, P, H), P is associated with measurable input signals (it could be designed Guassian white noise signals, for example, that used for rich system stimulation, but the time series is available to us), and H is associated with stochastic noise, of which we may only or may not know the distribution (e.g. mean, variance, etc.). Thus, to be consistent, we use (Q, H) to refer to the DSF with white noise as "inputs".

Theorem 2.14 (Hayden et al. (2016b)). *Two systems (A, B, C, D) and (A', B', C', D') under Assumptions 2.1- 2.6 with DSFs (Q, H) and (Q', H') have equal output spectral density*

$$\Phi(s) = G(s)G^*(s) = G'(s)G'^*(s)$$

if and only if $Q = Q'$ and $H = \pm H'$. Given $\Phi(s)$, there is therefore only one solution $(Q, \pm H)$ with minimum phase G.

If the transfer function $G(s)$ is NOT minimum phase, we no longer has the result of unique factorization (up to signs) of (Q, H) from output spectral density. Since we are not going to use this conclusion in our study, it is skipped here and one may see Hayden et al. (2016b) for more details.

Sparse network identifiability via compressed sensing

Hayden et al. (2016a) analyzes identifiability of the noise-free DSF (i.e. (Q, P)). It is assessing the uniqueness of Q and P under the following assumptions. It is manifested in the frequency domain, where $Y(s), U(s)$ are used to denote the Laplace transformation of $y(t), u(t)$, respectively.

Assumption 2.7. A set of m experiments is given, comprising known inputs U and outputs Y to $Y = QY + PU$.

Assumption 2.8. The number of experiments m is fewer than the number of manifest states p.

Assumption 2.9. The matrix P is square, diagonal and full rank.

Assumption 2.10. The rows of Q are k-sparse:

$$\|Q(i, :)\|_0 \leq k < p \quad \text{for } i = 1, \ldots, p$$

where k is a known parameter.

The problem of network reconstruction is essentially to seek sparse solution Q and diagonal P to

$$\begin{bmatrix} Y^T & U^T \end{bmatrix} \begin{bmatrix} Q^T \\ P^T \end{bmatrix} = Y^T. \tag{2.28}$$

Hayden et al. (2016a) presents conclusions on single input experiments: constructive solutions to sparse Q and identifiable entries of Q, as summarized in the following propositions. Here the single input experiment refers to that in each experiment only one input is applied such that U can be written as

$$U = \begin{bmatrix} U_1 \\ 0 \end{bmatrix}, \tag{2.29}$$

where U_1 is square, diagonal and of dimension $m \times m$. Partitioning Q and P accordingly, the reconstruction equation (2.28) can be written as

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} + \begin{bmatrix} P_{11} \\ 0 \end{bmatrix} U_1, \tag{2.30}$$

where P_{11} si square and diagonal.

Proposition 2.15 (Hayden et al. (2016a)). *A particular solution to the sparse network reconstruction problem* (2.28) *under Assumption 2.7- 2.10 and with single experiments* (2.29) *is given by*

$$\hat{Q} = \begin{bmatrix} \hat{Q}_{11} & 0 \\ \hat{Q}_{21} & 0 \end{bmatrix}, \quad \hat{P} = \begin{bmatrix} \hat{P}_{11} & 0 \\ 0 & 0 \end{bmatrix},$$

where $(\hat{Q}_{11}, \hat{P}_{11})$ *and* $\hat{Q}_{21}$ *are defined as*

$$\begin{aligned} \hat{Q}_{11} &= (I - D_{11})^{-1}(\bar{Q}_{11} - D_{11}), \\ \hat{P}_{11} &= (I - D_{11})^{-1} P_{11}, \\ \hat{Q}_{21} &= (I - Q_{22})^{-1} Q_{21}, \end{aligned}$$

and $\bar{Q}_{11} = Q_{11} + Q_{12}(I - Q_{22})^{-1} Q_{21}$, $D_{11} = \operatorname{diag}(\operatorname{diag}(\bar{Q}_{11}))$.

Assume single inputs of the form (2.29) have been applied and partition Q and $\hat{Q}$ of Proposition 2.15 as follows

$$Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}, \quad \hat{Q} = \begin{bmatrix} \hat{Q}_1 & 0 \end{bmatrix}, \quad \hat{Q}_1 \triangleq \begin{bmatrix} \hat{Q}_{11} \\ \hat{Q}_{21} \end{bmatrix},$$

such that Q_1 has the same dimension as $\hat{Q}_1$.

Proposition 2.16 (Hayden et al. (2016a)). *For every* $i \neq j$*, if* $\hat{Q}_1(i,j) \neq 0$*, then*

$$Q_1(i,j) \neq 0 \quad \text{or} \quad Q_2(i,k)\hat{Q}_{21}(k,j) \neq 0$$

for some k*. Else if* $\hat{Q}_1(i,j) = 0$*, then*

$$Q_1(i,j) = 0, \quad \text{and} \quad \sum_{k=1}^{p-m} Q_2(i,k)\hat{Q}_{21}(k,j) = 0.$$

2.3 Theory of Dynamical Structure Functions

2.3.1 Partial structure representations

We introduce network structures associated with two partial structure representations: the *subsystem structure* and the *signal structure*. Both of them can be use to visualize the networked systems. However, they actually deliver different information.

The subsystem structure is an easy way to understand and exploit the properties of interconnected systems, which has a rich history in control engineering. It decomposes large complex systems into relevant subsystems to manage representational complexity. Let us introduce the definition of subsystem structure used in Chetty and Warnick (2015).

Definition 2.17. The linear *subsystem structure* dynamics for the i-th subsystem is given by

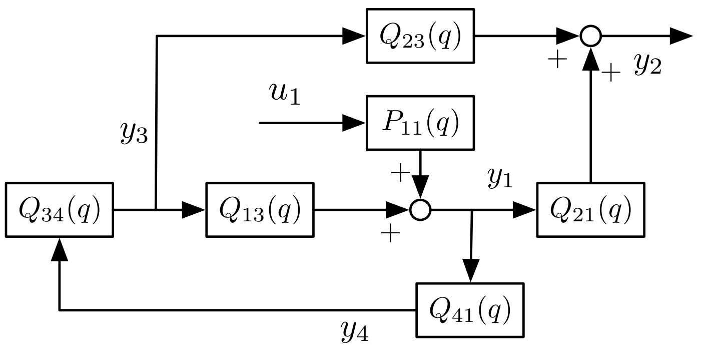
$$\dot{x}_i = A_i x_i + \sum_{j=1}^{n} W_{ij} x_j + B_i u_i$$

where $i = 1, \dots, q$, $j \neq i$, $x_i \in \mathbb{R}^{n_i}$, $u_i \in \mathbb{R}^{m_i}$, and A_i, W_{ij}, and B_i are of appropriate dimension.

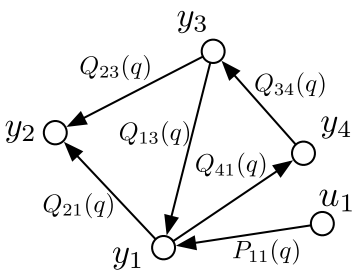
In the subsystem structure view of point, the example model (3.2) is visualized as Figure 2.4a. Strictly speaking, the DSFs should not be visualized in the subsystem structures, as explained later. Here we treat the model (3.2) as the matrix form of interconnected subsystems, instead of the DSF. The early studies of dynamic network by Prof. Paul van den Hof, e.g., Van den Hof et al. (2013), uses the subsystem structures.

The signal structure characterizes the open-loop causal dependencies among manifest variables (outputs). It reflects structural information about the internal closed-loop behavior of the system. The dynamic network (Definition 2.1) gives the graph-theoretical definition of signal structures for the DSF models. The signal structures of the DSFs describe system dynamics in terms of the causal dependencies among their manifest variables. Figure 2.4b shows the signal structure of the DSF model (3.2).

Both representations are dynamically equivalent, in that they each characterize the same manifest behavior of the system, but they are not structurally equivalent. It took us long time to understand of difference between two types of structure representations. The key point is whether it allows the share of hidden states. When subsystems are interconnected to form a new composite system, it deserves to be emphasized that variables internal to one subsystem are distinct from those internal to another. The DSF model allows the different elements in Q, P and H share hidden states. This is the reason why we should not visualize the DSFs in subsystem structures. Learning subsystem structure demands the identification of a partition of all system states, including hidden states that are not measured. On the



(a) subsystem structure



(b) signal structure

Figure 2.4. An example of the DSF model visualized in subsystem structures and signal structures.

contrary, learning signal structure, the DSFs, from data often requires much less *a priori* information about a system. Moreover, due to the same reason, the assumption of *known structures* for subsystem structures delivers more information than that for signal structures. The state-partitioning property of the subsystem structure potentially restricts the set of admissible realization much more severely than the signal structure. See Warnick (2015) for a strict discussion using the language of *behavioral systems* (Willems, 2007).

In the study of module network identification/identifiability, it seems the subsystem structure is a particularly convenient representation. One may see examples from (Van den Hof et al., 2013, 2017; Weerts et al., 2016, 2018a). It is easy to see this point by noticing that the identification of one network module is to identify this transfer function with a feedback controller that consists of all the other modules. However, for the identifiability study of the whole network, one has to be careful about the difference between these two structure representations.

2.3.2 Realization and structure degrees

The *definition* of the DSFs refers to the procedure (2.3)-(2.7) that derives the DSF from a given state-space model. And we call a state-space model is *consistent* with a DSF if this state-space model gives the DSF by the aforementioned definition. Moreover, we say a DSF is *consistent* with an MIMO transfer function if there exists a realization of the transfer function that is consistent of the DSF. Given a DSF, we have a unique transfer function that the DSF is consistent with (Goncalves and Warnick, 2008). A *realization* of the DSFs is the reverse of the definition of the DSFs, which is to find a state-space model that is consistent with the given DSF model. The relations of these representations can be illustrated by Figure 2.5.

Without loss of generality, let us consider deterministic LTI systems with C being of full row rank. Let $\Sigma \triangleq (A, B, C, D)$ denote a state-space model, $G(s)$ be an input-output transfer function, and $(Q(s), P(s))$ denote a DSF model. And we have $G = (I - Q)^{-1}P$, if $(I - Q)$ is invertible (e.g., Q is strictly proper). Before starting the topic on properties of

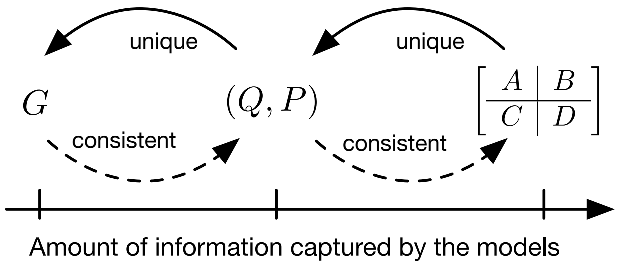

unique
unique
G
(Q, P)
$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$
consistent
consistent
Amount of information captured by the models

Figure 2.5. Hierarchical structure of data representations: transfer function, dynamical structure functions and state-space models. The solid arrows mean "A uniquely determines B", and the dashed arrows means "there exists at least one B that is consistent with A" (considering an example $A \to B$).

different types of realizations, we have to make sure the given DSFs or the signal structures, characterized by (Q, P), are realizable, which is the problem of *well-posedness* of DSFs. The conditions of well-posedness on DSFs are given as follows by Woodbury et al. (2017).

Theorem 2.18. *Consider the signal structure of an LTI system characterized by* $(Q(s), P(s))$*. This signal structure is well-posed if and only if* $(I - Q(\infty))$ *is invertible.*

It is clear that the well-posedness of DSFs is guaranteed for strictly proper Q (since $I - Q(\infty)$ is trivially equal to I). This is the reason why we neglect this issue in Goncalves and Warnick (2008). However, when we try to extend the definition of DSFs to include proper Q's, e.g., Section 2.1.3, we have to be particularly careful about the well-posedness issue.

Definition 2.19. We say that a realization Σ of $G(s)$ is *G-minimal* if Σ is a minimal realization of $G(s)$. We say that a realization Σ of $(Q(s), P(s))$ is *(Q, P)-minimal* if A of Σ has the smallest possible dimension. We call $\deg(\Sigma)$ the *structure degree* of $(Q(s), P(s))$, where Σ is (Q, P)-minimal and $\deg(\Sigma)$ denotes the dimension of A of the realization Σ.

As well known, the *McMillan degree* of $G(s)$ is equal to $\deg(\Sigma)$, where Σ is G-minimal. The minimal realization of $G(s)$ has been soundly studied in control theory and its construction is straightforward. On the contrary, the minimal realization of (Q, P) is particularly challenging. Yuan et al. (2015) presents an algorithm of minimal realization of (Q, P) when (Q, P) only has simple poles and does not have the same poles and zeros.

One may have noticed that there have been two types of realizations: realizations of (Q, P) and realizations of G. Here we will introduce another realization, based on the knowledge on the difference between subsystem structures and signal structures, presented in Section 2.3.1.

[7]The standard definition of minimal realization of MIMO transfer functions, e.g. (Zhou et al., 1996, p. 68).

Definition 2.20. We say a family of state-space models $\Sigma \triangleq \{\Sigma_i : i = 1, \ldots, n\}$ is a *subsystem realization* of (Q, P) if each Σ_i is a realization[8] of a nonzero element of Q or P, where n is the number of nonzero elements in Q and P. We define $\deg(\Sigma)$ by the sum of $\deg(\Sigma_i), i = 1, \ldots, n$. This realization Σ is called minimal, denoted by $(Q, P)_{\text{sub}}$*-minimal* if each Σ_i is a minimal realization.

With the knowledge on the McMillan degrees and minimal realizations of transfer functions, we know that $\deg(\Sigma)$ is equal to the sum of the McMillan degrees of each elements of Q and P, where Σ is the minimal subsystem realization of (Q, P).

We then would like to study the difference between the minimal degrees of the aforementioned three types of realizations. For example, is it true that a (Q, P)-minimal realization is also G-minimal, or $(Q, P)_{\text{sub}}$-minimal?

Proposition 2.21. *Given a DSF model (Q, P), let G be the corresponding transfer function. Let Σ_G be a minimal realization of G, Σ_{QP} be a minimal realization of (Q, P), and Σ_{sub} be a minimal subsystem realization of (Q, P). We have the following statement:*

$$\deg(\Sigma_G) \leq \deg(\Sigma_{QP}) \leq \deg(\Sigma_{\text{sub}}). \tag{2.31}$$

Proof. The proof of two inequalities is straightforward by noticing that Σ_{QP} is a realization of G, and Σ_{sub} is a realization of (Q, P). □

We call the difference $\deg(\Sigma_{\text{sub}}) - \deg(\Sigma_{QP})$ the number of *shared hidden states*.

Proposition 2.22. *Given a DSF model (Q, P), let G be the corresponding transfer function, and Σ_{QP}, Σ_G be the minimal realizations of (Q, P) and G, respectively. The difference $r_c = \deg(\Sigma_{QP}) - \deg(\Sigma_G)$ is equal to the number of uncontrollable states of Σ_{QP}.*

Proof. As well-known in control theory, a realization Σ_G is G-minimal if and only if Σ_G is controllable and observable. And Proposition 2.21 tells that $\deg(\Sigma_G) \leq \deg(\Sigma_{QP})$. The proof is done if we can show that any (Q, P)-minimal realization Σ_{QP} is observable. Suppose, on the contrary, that a (Q, P)-minimal Σ is not observable. Let the rank of the observability matrix be $k < n$. Since C is full row rank, we have $k \geq p$. The proof is then manifested in three cases.

[8] Since each nonzero element of Q or P is a SISO transfer function, the corresponding realization has been well-defined.

Case I: assume that $\Sigma = (A, B, C, D)$ has $C = \begin{bmatrix} I_{p\times p} & 0 \end{bmatrix}$ and is in the observable canonical decomposition, that is

$$\begin{aligned}
\dot{x} &= \left[\begin{array}{c|c:c} A_{11} & A_{12}^{o} & 0_{p\times(n-k)} \\ \hline A_{21}^{o} & A_{22}^{o} & 0_{(k-p)\times(n-k)} \\ \hdashline A_{21}^{\bar{o}} & A_{22}^{\bar{o}} & A_{\bar{o}} \end{array}\right] x + \left[\begin{array}{c} B_1 \\ \hline B_2^{o} \\ \hdashline B_2^{\bar{o}} \end{array}\right] u, \\
y &= \left[\begin{array}{c|c:c} I_{p\times p} & 0_{p\times(k-p)} & 0_{(n-k)\times(n-k)} \end{array}\right] x + Du.
\end{aligned} \tag{2.32}$$

For simplicity, we would not use subscripts to show the dimensions of I or 0 whenever it is clear. To show why Case I has the above partition, let the blocks be denoted by

$$A_{12} = \begin{bmatrix} A_{12}^{o} & 0 \end{bmatrix}, \qquad A_{21} = \begin{bmatrix} A_{21}^{o} \\ A_{21}^{\bar{o}} \end{bmatrix}, \qquad A_{22} = \begin{bmatrix} A_{22}^{o} & 0 \\ A_{22}^{\bar{o}} & A_{\bar{o}} \end{bmatrix}, \quad B_2 = \begin{bmatrix} B_2^{o} \\ B_2^{\bar{o}} \end{bmatrix},$$

$$A_o \triangleq \begin{bmatrix} A_{11} & A_{12}^{o} \\ A_{21}^{o} & A_{22}^{o} \end{bmatrix}, \quad A_{\bar{o}o} \triangleq \begin{bmatrix} A_{21}^{\bar{o}} & A_{22}^{\bar{o}} \end{bmatrix}, \quad B_o \triangleq \begin{bmatrix} B_1 \\ B_2^{o} \end{bmatrix}, \qquad C_o \triangleq \begin{bmatrix} I & 0_{p\times(k-p)} \end{bmatrix},$$

and $B_{\bar{o}} \triangleq B_2^{\bar{o}}$. It is easy to see that $\left(\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \begin{bmatrix} I & 0 \end{bmatrix}, D\right)$ is the canonical form as (2.3) to define the DSFs, and $\left(\begin{bmatrix} A_o & 0 \\ A_{\bar{o}o} & A_{\bar{o}} \end{bmatrix}, \begin{bmatrix} B_o \\ B_{\bar{o}} \end{bmatrix}, \begin{bmatrix} C_o & 0 \end{bmatrix}, D\right)$ is the observable canonical decomposition (here two 0's have different dimensions). The onus is to show the smaller dimensioned observable state space realization is also a realization of (Q, P), which contradicts the statement that Σ is (Q, P)-minimal. It is to show the following two equalities:

$$\begin{aligned}
A_{11} + \begin{bmatrix} A_{12}^{o} & 0 \end{bmatrix} \left(sI - \begin{bmatrix} A_{22}^{o} & 0 \\ A_{22}^{\bar{o}} & A_{\bar{o}} \end{bmatrix}\right)^{-1} \begin{bmatrix} A_{21}^{o} \\ A_{21}^{\bar{o}} \end{bmatrix} &= A_{11} + A_{12}^{o}(sI - A_{22}^{o})^{-1} A_{21}^{o}, \\
B_1 + \begin{bmatrix} A_{12}^{o} & 0 \end{bmatrix} \left(sI - \begin{bmatrix} A_{22}^{o} & 0 \\ A_{22}^{\bar{o}} & A_{\bar{o}} \end{bmatrix}\right)^{-1} \begin{bmatrix} B_2^{o} \\ B_2^{\bar{o}} \end{bmatrix} &= B_1 + A_{12}^{o}(sI - A_{22}^{o})^{-1} B_2^{o},
\end{aligned} \tag{2.33}$$

where 0's and I's have the appropriate dimensions as shown in (2.32). It is true by thinking of (2.33) as the calculation of corresponding transfer functions from the following two realizations and their observable realizations (of smaller dimensions):

$$\left[\begin{array}{c|c} \begin{bmatrix} A_{22}^{o} & 0 \\ A_{22}^{\bar{o}} & A_{\bar{o}} \end{bmatrix} & \begin{bmatrix} A_{21}^{o} \\ A_{21}^{\bar{o}} \end{bmatrix} \\ \hline \begin{bmatrix} A_{12}^{o} & 0 \end{bmatrix} & A_{11} \end{array}\right], \quad \left[\begin{array}{c|c} \begin{bmatrix} A_{22}^{o} & 0 \\ A_{22}^{\bar{o}} & A_{\bar{o}} \end{bmatrix} & \begin{bmatrix} B_2^{o} \\ B_2^{\bar{o}} \end{bmatrix} \\ \hline \begin{bmatrix} A_{12}^{o} & 0 \end{bmatrix} & B_1 \end{array}\right].$$

Case II: consider $\Sigma = (A, B, C, D)$ of the observable canonical decomposition, that is

$$\begin{aligned} \dot{x} &= \begin{bmatrix} A_o & 0_{k\times(n-k)} \\ A_{o\bar{o}} & A_{\bar{o}} \end{bmatrix} x + \begin{bmatrix} B_o \\ B_{\bar{o}} \end{bmatrix} u \\ y &= \begin{bmatrix} C_0 & 0_{p\times(n-k)} \end{bmatrix} x + Du. \end{aligned} \tag{2.34}$$

The idea to show this case is to convert it to Case I by a state transformation that preserves the same DSF. By Theorem 1 in Chetty and Warnick (2015), any state transformation of the form $T = \begin{bmatrix} C \\ E^T \end{bmatrix}$ is valid, where $E \in \mathbb{R}^{n\times(n-p)}$ is any basis of the null space of C, and $T^{-1} = \begin{bmatrix} C^\dagger & E \end{bmatrix}$, $C^\dagger \triangleq C^T(CC^T)^{-1}$. The key step to prove Case II is to construct such a T that converts Case II to Case I. We will prove the existence of such T's by construction. Let T be constructed as the following particular form

$$T = \left[\begin{array}{c} C \\ \hline E^T \end{array}\right] := \left[\begin{array}{cc} C_0 & 0_{p\times(n-k)} \\ \hline E_0^T & E_2^T \end{array}\right] := \left[\begin{array}{cc} C_0 & 0_{p\times(n-k)} \\ \hline C_{\hat{0}} & 0_{(k-p)\times(n-k)} \\ \hdashline \check{E}_0^T & \check{E}_2^T \end{array}\right] \triangleq \left[\begin{array}{cc} \bar{C}_0 & 0_{k\times(n-k)} \\ \hdashline \check{E}_0^T & \check{E}_2^T \end{array}\right]. \tag{2.35}$$

The A matrix after the transformation of T turns to be

$$\begin{aligned} TAT^{-1} &= \begin{bmatrix} \bar{C}_0 & 0 \\ \check{E}_0^T & \check{E}_2^T \end{bmatrix} \begin{bmatrix} A_o & 0 \\ A_{o\bar{o}} & A_{\bar{o}} \end{bmatrix} \begin{bmatrix} \bar{C}_0^\dagger & \check{E}_0 \\ 0 & \check{E}_2 \end{bmatrix} \\ &= \begin{bmatrix} \bar{C}_0 A_o \bar{C}_0^\dagger & \bar{C}_0 A_o \check{E}_0 \\ \check{E}_0^T A_o \bar{C}_0^\dagger + \check{E}_2^T A_{o\bar{o}} \bar{C}_0^\dagger & \times \end{bmatrix}, \end{aligned} \tag{2.36}$$

where $\bar{C}_0^\dagger \triangleq \bar{C}_0^T(\bar{C}_0\bar{C}_0^T)^{-1}$, and $\times$ in the partitioned matrix denotes the block that is not interesting in our analysis. To convert Case II to Case I, it requires that

$$CE = 0 \tag{2.37a}$$

$$\operatorname{rank}(E) = n - p \tag{2.37b}$$

$$\bar{C}_0 A_0 \check{E}_0 = 0. \tag{2.37c}$$

The equalities (2.37a) and (2.37b) come from the original requirements of T that preserves the same DSF. The equality (2.37c) guarantees that the A matrix keeps the form of observability canonical decomposition after the state transformation. The remaining is to show that there exists at least one solution E to (2.37). To close the proof, we provide a solution E to (2.37) by construction. Substituting (2.35), the equalities (2.37a) and (2.37c) yield that

$$\begin{aligned} C_0 C_{\hat{0}}^T = 0, &\qquad C_0 \check{E}_0 = 0, \\ C_0 A_0 \check{E}_0 = 0, &\quad C_{\hat{0}} A_0 \check{E}_0 = 0. \end{aligned} \tag{2.38}$$

By solving (2.38), it is easy to construct a T as follows to be a solution to (2.37):

- $\check{E}_0 = 0$,
- $C_{\hat{0}}$ is any basis of null space of C_0, and
- $\check{E}_2$ is any invertible matrix of dimension $((n-k) \times (n-k))$.

Case 3: consider any realization $\Sigma = (A, B, C, D)$ of (Q, P) with C being full row rank[9]. Let $\mathcal{O}$ be the observability matrix

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix},$$

and $\operatorname{rank}(\mathcal{O}) = n$. Noticing that $\operatorname{rank}(C) = p$, by Cayley-Hamilton theorem, any basis of the null space of C can be expressed as a linear combination of CA^k $(k = 1, \ldots, n-1)$. Hence the E in the state transformation T can be constructed in terms of CA^k, and T transforms the Σ into Case II. $\square$

The above results are summarized and illustrated by Figure 2.6.

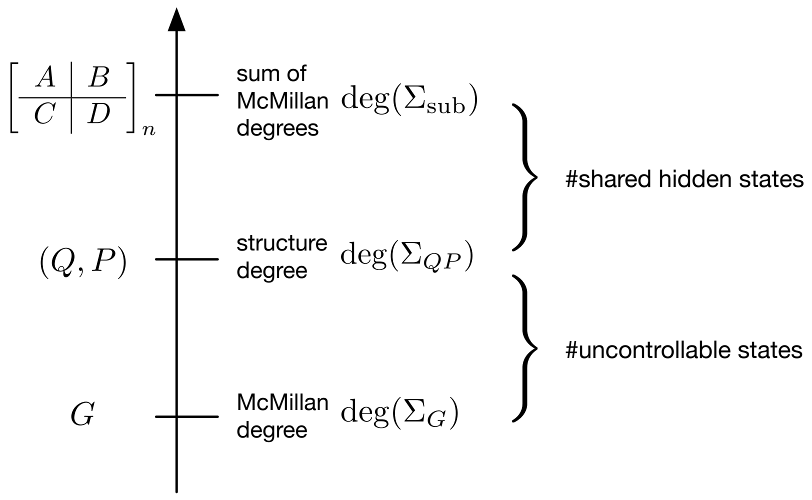


Figure 2.6. The degrees of minimal realizations of G, (Q, P) and the subsystems; and their relations.

2.3.3 Network stability

Without loss of generality, let us consider a deterministic continuous-time LTI system $\Sigma \triangleq (A, B, C, D)$ with full row rank C, which gives the signal structure $(Q(s), P(s))$ and

[9]Note that we cannot have C that is not full row rank but the realization leads to a strictly proper Q. Hence, it is not an additional constraint that C is full row rank.

the transfer function $G(s)$. Let the dimension of the state variable be n, that of the output variable be p and that of the input variable be m. Noticing that Q is strictly proper since the C is full row rank, the well-posedness of the signal structures (Q, P) is always guaranteed. And we know that $G(s) = D + C(sI - A)^{-1}B$ and $G(s) = (I - Q(s))^{-1}P(s)$. As well known in linear system theory, the system Σ is *internally stable* (also *uniformly exponentially stable*) if and only if all eigenvalues of A have negative real parts (i.e. $\mathrm{Re}(\lambda_i(A)) < 0, i = 1, \dots, n$). The system is *input-output stable*, or strictly speaking, *bounded-input, bounded-output stable* (BIBO stable), if the poles of $G(s)$ are in the open left half of the complex plane. The internal stability leads to the input-output stability, and the reverse is false. Similarly to Figure 2.5, which shows the hierarchy of three representations on information delivery, we would like to propose a definition of *network stability*, i.e. the stability of DSFs. To be specific, we expect the following, as illustrated in Figure 2.7: a) internal stability implies network stability, which implies input-output stability; b) input-output stability may not lead to network stability, which may not lead to internal stability.

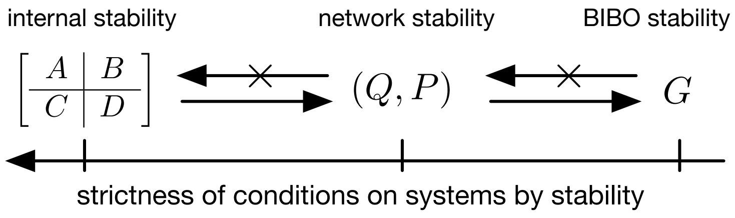

internal stability
network stability
BIBO stability
$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$
(Q, P)
G
strictness of conditions on systems by stability

Figure 2.7. Hierarchical structure of system stability and network stability. The arrow in $A \to B$ means A implies B; and the arrow attached with a cross means such an implication could be false.

Based on this principle, we propose the straightforward definition on network stability, and the propositions that show it satisfies the principle. One has to know that there could be alternative definitions of network stability, which could be either close to internal stability or to input-output stability. The phrase "being close to X stability" means the fewer additional conditions together with network stability guarantee "X stability". Before presenting Definition 2.24, we first quickly review a few properties of the DSFs.

Proposition 2.23. *Given a DSF $(Q(s), P(s))$, we have the following well known properties:*

- *each element of $Q(s)$, except zeros, is a strictly proper real-rational transfer function;*
- *each element of $P(s)$ is a proper real-rational transfer function (when consider a system Σ with $D = 0$, they are also strictly proper);*
- *$Q(s)$ is square and $(I - Q(s))$ is invertible.*

There are multiple ways to define network stability to satisfy the principle illustrated in Figure 2.7. Here we present two feasible definitions, one of which is closer to the internal stability and the other is close to the input-output stability.

Definition 2.24 (network stability I). Consider the signal structure of an LTI system characterized by the DSF $(Q(s), P(s))$. The signal structure is stable (or, the system is *network stable*) if

- all poles of $Q(s)$ and $P(s)$ have negative real parts, and
- all transmission zeros of $(I - Q(s))$ have negative real parts.

In the following, we will prove that such a definition of network stability satisfies the aforementioned principle.

Lemma 2.25. *Considering a stable system $\Sigma = (A, B, C, D)$, all transmission zeros of $(sI - W(s))$ have negative real parts, where $W(s)$ is defined in (2.5).*

Proof. Since A is stable, $\hat{A} = TAT^{-1}$ is stable. Letting $M = \lambda I_A - \hat{A}$, the equality $\det(M) = \det(M_{22}) \det(M/_{22})$ implies that the solutions to $\det(M/_{22}) = 0$ have negative real parts, where $M_{22} = \lambda I_2 - \hat{A}_{22}$, $M/_{22} = (\lambda I_1 - \hat{A}_{11}) - \hat{A}_{12}(\lambda I_2 - \hat{A}_{22})^{-1}\hat{A}_{21}$ and I_A, I_1, I_2 are the identity matrices of appropriate dimensions. Notice that the algebraic equation $\det(M/_{22}) = 0$ w.r.t. $\lambda \in \mathbb{C}$, whose all solutions have negative real parts due to stable A, is the same as $\det(sI - W(s)) = 0$ w.r.t. $s \in \mathbb{C}$. Moreover, since A is stable, $(sI - W(s))$ has full normal rank[10]. Hence, to calculate the transmission zeros of $(sI - W(s))$, by Lemma 3.29 in Zhou et al. (1996) (which is also used in other texts as a definition of transmission zeros), we simply solve the algebraic equation $\det(M/_{22}) = 0$ w.r.t. λ's, which all have negative real parts. □

Proposition 2.26. *If a system is internal stable, it is network stable.*

Proof. The internal stability implies all eigenvalues of A have negative real parts, so does that of $\hat{A}$ in (2.3). It then yields that all eigenvalues of $\hat{A}_{22}$ have negative real parts. Indeed, letting $M = \lambda I - \hat{A}$, the equality $\det(M) = \det(M_{22}) \det(M/_{22})$ implies that all eigenvalues of $\hat{A}_{22}$ appear as the eigenvalues of $\hat{A}$, where $M_{22} = \lambda I_2 - \hat{A}_{22}$, $M/_{22} = (\lambda I_1 - \hat{A}_{11}) - \hat{A}_{12}(\lambda I_2 - \hat{A}_{22})^{-1}\hat{A}_{21}$ and I_1, I_2 are the identity matrices of appropriate dimensions. It then yields that the transfer matrices $W(s)$ and $V(s)$ given in (2.5) are stable and hence network stability, by considering $W(s)$ as the input-output transfer function of the realization $(\hat{A}_{22}, \hat{A}_{21}, \hat{A}_{12}, \hat{A}_{11})$, and $V(s)$ as that of $(\hat{A}_{22}, \hat{B}_2, \hat{A}_{12}, \hat{B}_1)$. To show $Q(s)$ and $P(s)$ being stable, it requires in addition that $(sI - D_W)^{-1}$ used in (2.6) is stable, which has been guaranteed by Lemma 2.25. The remaining is to prove the second condition. We rewrite $(I - Q)$ into $(sI - D_W)^{-1}(sI - D_W)[I - (sI - D_W)^{-1}(W - D_W)] = (sI - D_W)^{-1}(sI - W)$. Lemma 2.25 tells that all transmission zeros of $(sI - W)$ have negative real parts. And the

[10]See, for example, (Zhou et al., 1996, chap. 3.11) for the definition of *normal rank*.

diagonal matrix $(sI - D_W)$ is stable. Hence, it follows that all transmission zeros of $(I - Q)$ have negative real parts. $\square$

Proposition 2.27. *If a system is network stable, it is input-output stable.*

Proof. Consider $G(s) = [I - Q(s)]^{-1}P(s)$. Network stability guarantees that $P(s)$ is stable and all transmission zeros of $(I - Q(s))$ have negative real parts. Recalling that $(I - Q(s))$ is invertible, $G(s)$ is hence guaranteed to be stable. Indeed, consider the McMillan form of the invertible $(I - Q(s))$

$$U(s)(I - Q(s))V(s) = M(s) := \begin{bmatrix} \frac{\alpha_1(s)}{\beta_1(s)} & & \\ & \ddots & \\ & & \frac{\alpha_p(s)}{\beta_p(s)} \end{bmatrix},$$

where $U(s)$ and $V(s)$ are unimodular matrices in $\mathbb{R}(s)$. Then $V^{-1}(s)(I - Q(s))^{-1}U^{-1}(s) = M^{-1}(s)$ is the McMillan form of $(I-Q(s))^{-1}$, and the poles of $(I-Q(s))^{-1}$ are the transmission zeros of $(I - Q(s))$, which have negative real parts. $\square$

Definition 2.28 (network stability II). Consider the signal structure of an LTI system characterized by the DSF $(Q(s), P(s))$. The signal structure is stable (or, the system is *network stable*) if $P(s), [I - Q(s)]^{-1} \in \mathcal{RH}_\infty$, i.e. they are proper and real-rational stable transfer matrices.

Provided with the proofs above for Definition 2.24, it is easy to see that Proposition 2.26 and 2.27 also hold for Definition 2.28. First, it is trivial to see that Proposition 2.27 holds by noticing that $G = (I - Q)^{-1}P$. Considering Proposition 2.26, $P(s) \in \mathcal{RH}_\infty$ has been shown by the proof of Proposition 2.26 given before. And $[I - Q(s)]^{-1} \in \mathcal{RH}_\infty$ is shown by combining the reasoning given in the proofs of Proposition 2.26 and 2.27.

Remark 2.3. These two definitions are not equivalent. The conditions used in Definition 2.24 is more strict than and yield the ones used in Definition 2.28. This is why we state that Definition 2.24 is "closer" to the internal stability. We can also see that the bounded-input bounded-output data is not necessary from a networked system that each element in $Q(s)$ is stable.

2.3.4 Information flow

In the following we will explain how the DSF deals with latent variables and defines the path diagrams. The preservation of the input-output behaviors in the DSF is obvious considering the algebraic procedure on deriving the DSF from state-space realizations. We consider the continuous-time DSF (2.2) and its derivation procedure (2.4)-(2.7). Here we assume $K = 0, C = \begin{bmatrix} I & 0 \end{bmatrix}$, and $D = 0$. The same reasoning can be applied to the discrete-time DSF without additional difficulties.

Supposing $A_{22}(s)$ is a bounded operator and $\|A_{22}(s)\| \leq 1$, we have that $(I - A_{22}(s))^{-1}$ is well defined in the *Banach* space $\mathscr{L}(Z, Z)$, where Z denotes the set of $z(t)$ with any fixed $t \in \mathbb{R}_+$ (see (Reed and Simon, 1972, chap. VI) for more details). Slightly abusing notations, we use the indices of x to define the index set of the path-diagram (here we focus on Q), i.e. $V \coloneqq \{1, \ldots, p\}$. Let Λ_Z be the indices of hidden states, i.e. $\Lambda_Z = \{1, \ldots, n-p\}$. Referring to (2.3), y_i $(i \in V)$ is equivalently x_i, while z_i $(i \in \Lambda_Z)$ corresponds to x_{i+p}. Note that $\forall a, b \in V$, $a \to b \in E \Leftrightarrow Q_{b,a} \neq 0$, which is equivalent to $W_{b,a} \neq 0$. By the definition of W,

$$\begin{aligned} W_{b,a} &= A_{11}(b,a) + A_{12}(b,:)(I_2 - A_{22})^{-1} A_{21}(:,a) \\ &\triangleq A_{11}(b,a) + \bar{A}_{11}(b,a). \end{aligned} \tag{2.39}$$

Consider $(I - A_{22})^{-1} \triangleq \bar{A}_{22}$ by its *Neumann* series expansion, assuming $\|A_{22}(s)\| < 1$ is satisfied, yielding

$$\begin{aligned} \bar{A}_{22}(i,j) = \delta(i,j) + A_{22}(i,j) + \textstyle\sum_{k=1}^{n-p} A_{22}(i,k) A_{22}(k,j) \\ + \textstyle\sum_{k_1,k_2=1}^{n-p} A_{22}(i,k_1) A_{22}(k_1,k_2) A_{22}(k_2,j) + \cdots \end{aligned} \tag{2.40}$$

where $\delta(i,j) = 1$, if $i = j$; otherwise, $\delta(i,j) = 0$. The appended proof at the end of this section shows that (2.40) has finite terms in the sense of a quotient set, where each element represents a path. Recall that $A(i,j) \neq 0$ implies a directed path $x_i \leftarrow x_j$. Thus, each non-zero item of summation in (2.40) indicates a directed path $z_i \leftarrow z_{k_1} \leftarrow \cdots \leftarrow z_{k_{n-1}} \leftarrow z_j$, where z_k $(k \in \Lambda_Z)$ (i.e. x_{k+p}) are hidden states. Substituting (2.40) into (2.39) yields

$$\bar{A}_{11}(b,a) = \sum_{i,j \in \Lambda_Z} A_{12}(b,i) \bar{A}_{22}(i,j) A_{21}(j,a) \triangleq \sum_{i,j \in \Lambda_Z} \bar{A}_{11}^{ba}(i,j), \tag{2.41}$$

in which each non-zero $\bar{A}_{11}^{ba}(i,j)$ indicates a path $y_b \leftarrow z_i \leftarrow \cdots \leftarrow z_j \leftarrow y_a$. In particular, when $i = j$, the non-zero item indicates a path $y_b \leftarrow z_i \leftarrow y_a$. However, it is possible in theory that $\bar{A}_{11}(b,a) = 0$ with non-zero items $\bar{A}_{11}^{ba}(i,j)$, which we refer to by *exact cancellation*. Moreover, $A_{11}(b,a) \neq 0$ indicates a direct edge $b \leftarrow a$. It follows that

- $W_{b,a} \neq 0 \Rightarrow$ there exists at least one directed path $\pi_{ba} \colon y_b \leftarrow z_{k_1} \leftarrow \ldots \leftarrow z_{k_{n-1}} \leftarrow y_a$, with vertices $z_{k_1}, \ldots, z_{k_{n-1}} \in V_F \setminus V$, or $a \to b$.
- there exists at least one directed path $\pi_{ba} \colon y_b \leftarrow z_{k_1} \leftarrow \ldots \leftarrow z_{k_{n-1}} \leftarrow y_a$, with vertices $z_{k_1}, \ldots, z_{k_{n-1}} \in V_F \setminus V$, or $a \to b \Rightarrow W_{b,a} \neq 0$ almost surely.

One can see that the DSF is a proper mathematical representation (or, parametric model) that accumulates the information flow along directed paths via latent variables, which successfully resolves all counterexamples addressed in Eichler (2006).

Remark 2.4. The range of applicability deserves to be emphasized. Granger Causality is defined for and works ONLY with weakly stationary processes, in which "*the stochastic nature of the variables and the direction of the flow of time will be central features*" (Granger, 1969). The DSF is defined for LTI systems in general. However, extra studies may be required to guarantee the uniqueness of identification from data.

Considering hidden states (or latent variables), i.e. $V_F \setminus V$, Granger Causality cannot correctly deal with all of them, which leads to *spurious causality* (see the definition in Hsiao (1982) and examples in Eichler (2006)). In the level of definitions, the DSF successfully deals with latent variables, and gives a uniform definition of causal networks for LTI systems. It correctly deals with latent variables concerning the information flow described in Eichler (2006).

Appendix of proof

This section is to show that "the Neumann series of $(I_2 - A_{22}(q))^{-1}$ represents finite paths in the sense of a quotient set, if the inverse exists". Consider (2.40), which is the (i, j)-th element of $(I_2 - A_{22})^{-1}$. Let $S^{ij}_{A_{22}}$ be the set of the terms in the right-hand side of (2.40) (i.e. $S^{ij}_{A_{22}} := \{\delta(i,j),\ A_{22}(i,j),\ \sum_{k=1}^{n-p} A_{22}(i,k)A_{22}(k,j), ...\}$), which is a countable set (recall that the union of a countable collection of countable sets is still countable). In graph theory, each element in $S^{ij}_{A_{22}}$ represents a path $\pi_N := \langle e_{i,k_1}, e_{k_1,k_2}, ..., e_{k_N,j}\rangle$ from z_i to z_j. A *loop* is a closed path that originates and terminates on the same vertex, with no vertex being met twice along the path. Let $\mathcal{L}_{\pi_N}$ be a set of all loops *on* the path π_N, where a loop l *on* π_N means $l \subset \pi_N$. (Refer to Mason (1953, 1956) for more details on loops.)

Define an *equivalence relation* $\sim$ on $S^{ij}_{A_{22}}$ as a binary relation satisfying: $\forall p_1, p_2 \in S^{ij}_{A_{22}}$ and $p_1 \neq p_2$, $p_1 \sim p_2$ if and only if $p_1 \setminus \bigcup \mathcal{L}_{\pi_1} = p_2 \setminus \bigcup \mathcal{L}_{\pi_2}$. The *equivalence classes* of an element p in $S^{ij}_{A_{22}}$, denoted as $[p]$, is defined as $[p] := \{\pi \in S^{ij}_{A_{22}} : p \sim \pi\}$. Then we have the *quotient set* of $S^{ij}_{A_{22}}$ defined as a set of equivalence classes on $S^{ij}_{A_{22}}$, denoted by $S^{ij}_{A_{22}}/_\sim$. Each equivalence class in $S^{ij}_{A_{22}}/_\sim$ represents a path π_k of length k from i to j by overlooking the loops. Then the only thing left is to show $S^{ij}_{A_{22}}/_\sim$ is a finite set. To determine the cardinality of $S^{ij}_{A_{22}}/_\sim$, it is equivalent to count the number of paths in $S^{ij}_{A_{22}}$ that consist of distinct edges only. Therefore,

$$\text{card}(S^{ij}_{A_{22}}/_\sim) \leq \sum_{k=1}^{n-p-2} k! \binom{n-p-2}{k} + 1.$$

Here adding 1 is to include the shortest path $\pi_0 = e_{ij}$, if existing.

Chapter 3

Heterogeneous Datasets

The heterogeneity of datasets is widely observed but mostly ignored due to difficulties in data analysis. In practice, datasets in multiple experiments are inevitably subjected to different experimental conditions, e.g. perturbations in parameters, disturbance or noise. Nevertheless, the individuals in experiment repeats must have common properties that are focused on in experiments. The essence could the interconnection structure, e.g. interactions between genes, or communication between neurons, which are consistent over experiment repetitions. In the network reconstruction problems, we assume such common properties are represented by Boolean network structure. For instance, the healthy individuals in the control group have the same genetic regulatory mechanism, even though they might differ in dynamics. The study in this chapter discusses how to integrate all available datasets, which are heterogeneous, to perform network reconstruction. The method proposed in the chapter allows the system parameters to be fairly different, as long as the network topology keeps consistent over replica.

The methodology in this chapter is organized in modules, illustrated in Figure 3.1: 1) derivation of regression problems for network reconstruction (Section 3); 2) dealing with heterogeneity data (Section 4); 3) solvers to the target optimization problem (Section 5). The paper is closed by benchmark studies on random sparse network reconstruction.

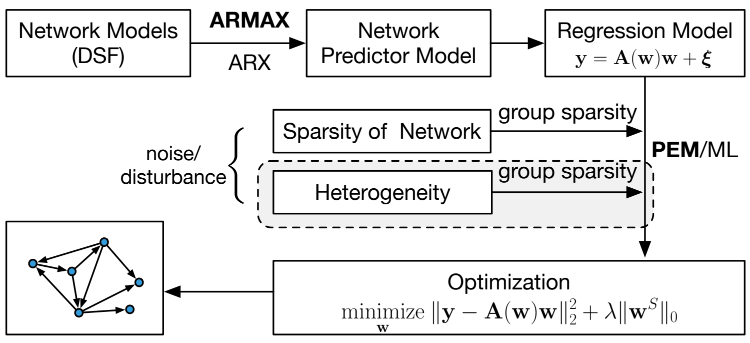

Network Models (DSF)
ARMAX
ARX
Network Predictor Model
Regression Model $\mathbf{y} = \mathbf{A}(\mathbf{w})\mathbf{w} + \boldsymbol{\xi}$
Sparsity of Network
group sparsity
PEM/ML
noise/ disturbance
Heterogeneity
group sparsity
Optimization $\underset{\mathbf{w}}{\text{minimize}}\ \|\mathbf{y} - \mathbf{A}(\mathbf{w})\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}^S\|_0$

Figure 3.1. An overview of the network reconstruction method.

3.1 Problem Description

Let $Y \triangleq \{y(t), t \in \mathbb{Z}\}$, $U \triangleq \{u(t), t \in \mathbb{Z}\}$ be multivariate time series of dimension p and m, respectively, where the elements could be deterministic ($y(t) \in \mathbb{R}^p$, $u(t) \in \mathbb{R}^m$) or be real-valued random vectors defined on probability spaces $(\Omega, \mathscr{F}, \mathbb{P})$. We usually assume that $u(t)$ is deterministic in practice, which is interpreted as controlled input signals.

3.1.1 Linear dynamic networks

Consider the network model for LTI systems, called *dynamical structure function* (DSF) Chetty and Warnick (2015); Goncalves and Warnick (2008),

$$y(t) = Q(q)y(t) + P(q)u(t) + H(q)e(t), \tag{3.1}$$

where $y(t) = [y_1(t), \dots, y_p(t)]^T$, $u(t) = [u_1(t), \dots, u_m(t)]^T$, a p-variate i.i.d. $e(t) = [e_1(t), \dots, e_p(t)]^T$ with zero mean and identity covariance matrix,

$$\begin{aligned} Q(q) &= [Q_{ij}(q)]_{p\times p}, & Q_{ii}(q) &= 0, \ \forall i, \\ P(q) &= [P_{ij}(q)]_{p\times m}, & H(q) &= [H_{ij}(q)]_{p\times p} \end{aligned}$$

$Q_{ij}(q), P_{ij}(q), H_{ij}(q)$ are single-input-single-output (SISO) real-rational transfer functions, and q is the forward-shift operator, i.e. $qy(t) = y(t+1)$, $q^{-1}y(t) = y(t-1)$. Here $Q_{ij}(q)$ ($i \neq j$) are strictly proper, and $P_{ij}, H_{ii}(q)$ are proper. See Chapter 2 for more on DSFs.

In regard to *networks* in this paper, we use the definition used in Yue et al. (2017). Let $\mathcal{G} = (V, E)$ be a digraph, where the vertex set $V = \{y_1, \dots, y_p, u_1, \dots, u_m, e_1, \dots, e_p\}$, and the directed edge set E is defined by

- $(y_j, y_i) \in E \Leftrightarrow Q_{ij}(q) \neq 0$,
- $(e_l, y_i) \in E \Leftrightarrow H_{il}(q) \neq 0$,
- $(u_k, y_i) \in E \Leftrightarrow P_{ik}(q) \neq 0$,
- $(y_i, u_k) \notin E$, $(y_i, e_l) \notin E$,

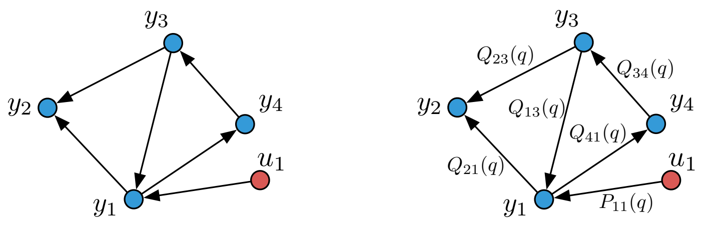
for all i, j, k and l. Let f be a map defined as

$$\begin{aligned} f: \quad & E \to S_{\mathrm{TF}} \\ & (y_j, y_i) \mapsto Q_{ij}(q) \ \text{ or } \ (u_k, y_i) \mapsto P_{ik}(q) \ \text{ or } \ (e_l, y_i) \mapsto H_{il}(q), \end{aligned}$$

where S_{TF} is a subset of single-input-single-output (SISO) proper rational transfer functions. We call the tuple $\mathcal{N} := (\mathcal{G}, f)$ a (linear) *dynamic network*, f the *capacity function* of $\mathcal{N}$, and $\mathcal{G}$ the *underlying digraph* of $\mathcal{N}$, which is also called (linear) *Boolean dynamic network*. For example, the dynamic network of $y = Qy + Pu$ with

$$Q = \begin{bmatrix} 0 & 0 & Q_{13} & 0 \\ Q_{21} & 0 & Q_{23} & 0 \\ 0 & 0 & 0 & Q_{34} \\ Q_{41} & 0 & 0 & 0 \end{bmatrix}, \quad P = \begin{bmatrix} P_{11} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.2}$$

is given as Figure 3.2. In applications the arcs from u or e to y may not be interesting. One could modify the definition by removing the corresponding terms from V and E.



(a) Boolean dynamic network (b) dynamic network

Figure 3.2. An example of dynamic networks for given DSFs.

To guarantee network identifiability, i.e. unique determination of DSFs from input-output transfer functions, we need to assume either $H(q)$ or $P(q)$ being square and diagonal, e.g. see Goncalves and Warnick (2008); Hayden et al. (2016b). Here we take the diagonal H as an example, and we further assume all assumptions in Hayden et al. (2016b) are satisfied.

Assumption 3.1. The matrix H is square, diagonal and full rank.

Alternatively, we could use general H but restrict $P(q)$ to be square and diagonal, and the forthcoming proposed method can be easily modified correspondingly. A square and diagonal P presents a practical way in experiments to guarantee network identifiability, i.e. perturbing each measured variable. However, in biological applications, we mostly need to kill cells to perform measurements. This leads to an issue that the underlying dynamics of cells in different experiments may not be perfectly identical, which is again the problem targeted in this paper. In practice, we may not afford such experiments and have to resort to assumptions on noise models (i.e. diagonal H) for network identifiability.

Remark 3.1. The DSFs with square diagonal $H(q)$ is fairly special, which can be equivalently written as $y(t) = H^{-1}(Q + H - I)y(t) + H^{-1}Pu(t) + e(t) \triangleq T_y(q)y(t) + T_u(q)u(t) + e(t)$. It is easy to see that $T_y(q)$ and $T_u(q)$ are proper transfer functions, and that $T_y(q)$ shares the same topology as $Q(q)$ and $T_u(q)$ shares the same topology as $P(q)$ since $H(q)$ is diagonal. Once we have T_y, T_u, the Q, P and H can be easily calculated from T_y, T_u, noticing that the diagonal of T_y is the addictive inverse of H. Therefore, the network reconstruction problems can be thought of as typical system identification problems with sparsity imposed on T_y, T_u.

Any identification methods with sparse input selection can be applied, e.g. the popular kernel methods (Pillonetto et al., 2011; Pillonetto and De Nicolao, 2010) with sparsity enhancement (Chiuso and Pillonetto, 2012). Nevertheless, a more general framework is used in the following sections such that the proposed method is not restricted to square diagonal H's.

3.1.2 Network reconstruction from multiple experiments

Now consider multiple experiments. Let $\{Y^{[c]}, U^{[c]}\}_{c=1,\dots,C}$ be measurements from C experiments. $\mathcal{N}((Q,P,H))$ denotes the dynamic network $\mathcal{N}$ determined by (Q,P,H), and $\mathcal{G}((Q,P,H))$ denotes the corresponding Boolean dynamic network. The governing model (3.1) could be different in each experiment, denoted by $(Q,P,H)^{[c]}, c = 1,\dots,C$. In addition, $\mathcal{N}^0$ denotes a fixed dynamic network, and $\mathcal{G}^0$ a fixed Boolean dynamic network. The datasets $\{Y^{[c]}, U^{[c]}\}_{c=1,\dots,C}$ are called *homogeneous*, if $\mathcal{N}((Q,P,H)^{[c]}) \equiv \mathcal{N}^0, \forall c$, i.e. the measurements in multiple experiments are from the same dynamic network. And the datasets $\{Y^{[c]}, U^{[c]}\}_{c=1,\dots,C}$ are said to be *heterogeneous*, if $\mathcal{G}((Q,P,H)^{[c]}) \equiv \mathcal{G}_0, \forall c$ but $\mathcal{N}((Q,P,H)^{[c]})$ are different between certain $c \in \{1,\dots,C\}$. Considering heterogeneity, additional constraints are demanded, given in Assumption 3.2, to present common features shared by systems in different experiments. For example, even though biological individuals are different in nature, if they are all healthy or subject to the same gene-level operations, it is fair to assume they have the same genetic regularization.

Assumption 3.2. The underlying systems in multiple experiments, which provide $\{Y^{[c]}, U^{[c]}\}_{c=1,\dots,C}$, satisfy that $\mathcal{G}((Q,P,H)^{[c]}) \equiv \mathcal{G}_0$ for any $c = 1,\dots,C$.

The problem is to develop methods to infer the dynamic Boolean network using the datasets from multiple experiments satisfying Assumption 3.2. In particular, we focus on the heterogeneous case.

3.2 Network model structures

This section shows a standard procedure to parametrize network models and derive corresponding regression problems. It can be applied to any type of time series models, such FIR, ARARX, ARARMAX, etc. We use ARMAX as an example in this paper.

3.2.1 ARMAX model structure

Consider the network *model description* of (3.1) for system identification

$$y(t) = Q(q,\theta)y(t) + P(q,\theta)u(t) + H(q,\theta)e(t), \tag{3.3}$$

where θ is the model parameter. Its element-wise form is

$$y_i(t) = \sum_{j=1}^{p} Q_{ij}(q,\theta)y_j(t) + \sum_{k=1}^{m} P_{ik}(q,\theta)u_k(t) + H_{ii}(q,\theta)e_i(t). \tag{3.4}$$

We introduce ARMAX model for (3.4), which is given by

$$A_i(q)y_i(t) = \sum_{j=1,j\neq i}^{p} B_{ij}^y(q)y_j(t) + \sum_{k=1}^{m} B_{ik}^u(q)u_k(t) + C_{ii}(q)e_i(t),$$

where

$$\begin{array}{rcrcl} A_i(q) & = & 1+a_{i1}\,q^{-1}+\cdots+ & a_{in_i^a}\,q^{-n_i^a}, \\ B_{ij}^y(q) & = & b_{ij1}^y\,q^{-1}+\cdots+ & b_{ijn_{ij}^{by}}^y\,q^{-n_{ij}^{by}}, \\ B_{ij}^u(q) & = & b_{ij1}^u\,q^{-1}+\cdots+ & b_{ijn_{ij}^{bu}}^u\,q^{-n_{ij}^{bu}}, \\ C_i(q) & = & 1+c_{i1}\,q^{-1}+\cdots+ & c_{in_i^c}\,q^{-n_i^c}. \end{array}$$

Hence the ARMAX model for (3.3) is

$$A(q)y(t) = B^y(q)y(t) + B^u(q)u(t) + C(q)e(t), \tag{3.5}$$

where

$$\begin{aligned} A &\triangleq \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_p \end{bmatrix}, \quad B^y \triangleq \begin{bmatrix} 0 & B_{12}^y & \cdots & B_{1p}^y \\ B_{21}^y & 0 & \dots & B_{2p}^y \\ \vdots & \vdots & \ddots & \vdots \\ B_{p1}^y & B_{p2}^y & \cdots & 0 \end{bmatrix}, \\ C &\triangleq \begin{bmatrix} C_1 & & \\ & \ddots & \\ & & C_p \end{bmatrix}, \quad B^u \triangleq \begin{bmatrix} B_{11}^u & B_{12}^u & \cdots & B_{1m}^u \\ B_{21}^u & B_{22}^u & \dots & B_{2m}^u \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1}^u & B_{m2}^u & \cdots & B_{mm}^u \end{bmatrix}. \end{aligned} \tag{3.6}$$

It is easy to see the following relations

$$Q(q,\theta) = A^{-1}B^y,\ P(q,\theta) = A^{-1}B^u,\ H(q,\theta) = A^{-1}C. \tag{3.7}$$

Remark 3.2. The model (3.5) simplifies to ARX if $C(q) \equiv I$, and it turns to an FIR model with $A(q) \equiv I$ in addition. Moreover, the orders $n_i^a, n_{ij}^{by}, n_{ij}^{bu}, n_i^c$ could be set to different values. However, in practice, we could start with the same values over different i and j; and tune them separately if necessary.

3.2.2 Network predictor model

Considering network model (3.1) and noticing that $[I - Q(q)]$ is invertible, we have $y(t) = (I-Q)^{-1}Pu(t) + (I-Q)^{-1}He(t) \triangleq G_u u(t) + G_e e(t)$. We refer to (Ljung, 1999, pp. 70) for the one-step-ahead prediction of y, $\hat{y}(t|t-1) = G_e^{-1}G_u u(t) + (I - G_e^{-1})y(t)$, and thus the network *predictor model* of (3.3) is given by $\hat{y}(t|t-1) = H^{-1}Pu(t) + H^{-1}(Q + H - I)y(t)$. The one-step-ahead predictor of the ARMAX model follows by substituting (3.7)

$$\hat{y}(t|\theta) = C^{-1}B^u u(t) + (C^{-1}B^y + I - C^{-1}A)y(t), \tag{3.8}$$

where $\hat{y}(t|t-1) \triangleq \hat{y}(t|\theta)$ to emphasize the dependency on model parameters θ.

3.2.3 Regression forms

Rewriting (3.8) and adding $[I - C(q)]\hat{y}(t|\theta)$ to both sides, it yields that

$$\hat{y}(t|\theta) = B^u(q)u(t) + \big[B^y(q) - \big(A(q) - I\big)\big]y(t) + \big(C(q) - I\big)\big[y(t) - \hat{y}(t|\theta)\big]. \tag{3.9}$$

To formulate a regression form, let us introduce the prediction error $\varepsilon(t|\theta) := y(t) - \hat{y}(t|\theta)$, and consider the prediction of the i-th output $y_i(t)$

$$\hat{y}_i(t|\theta) = \bar{B}_i^u(q)u(t) + \big[\bar{B}_i^y(q) - \big(\bar{A}_i(q) - \bar{I}_i\big)\big]y(t) + \big(C_i(q) - \bar{I}_i\big)\big[y_i(t) - \hat{y}_i(t|\theta)\big], \tag{3.10}$$

where $\bar{A}_i, \bar{B}_i^y, \bar{B}_i^u$, $\bar{I}_i$ are the corresponding i-th rows of A, B^y, B^u and I. Provided with

$$\begin{aligned}\varphi(t,\theta_i) \triangleq \big[\; & y_1(t-1)\ \dots\ y_1(t-n_{i1}^{by})\ \dots -y_i(t-1)\ \dots -y_i(t-n_i^{a})\dots \\ & y_p(t-1)\ \dots\ y_p(t-n_{ip}^{by}) \\ & u_1(t-1)\ \dots\ u_1(t-n_{i1}^{bu})\ \dots\ \ u_i(t-1)\ \dots\ \ u_i(t-n_{ii}^{bu})\dots \\ & u_m(t-1)\ \dots u_m(t-n_{im}^{bu}) \\ & \varepsilon_i(t-1)\ \dots\ \varepsilon_i(t-n_i^{c})\ \big]^T\end{aligned} \tag{3.11}$$

and

$$\begin{aligned}\theta_i \triangleq \big[\, & \boxed{\bar{b}_{i11}^{y}\ \cdots\ \bar{b}_{i1n_{i1}^{by}}^{y}}\ \cdots\ \boxed{a_{i1}\ \cdots\ a_{in_i^a}}\ \cdots\ \boxed{\bar{b}_{ip1}^{y}\ \cdots\ \bar{b}_{ipn_{ip}^{by}}^{y}} \\ & \boxed{\bar{b}_{i11}^{u}\ \cdots\ \bar{b}_{i1n_{i1}^{bu}}^{u}}\ \cdots\ \boxed{\bar{b}_{ii1}^{u}\ \cdots\ \bar{b}_{iin_{ii}^{bu}}^{u}}\ \cdots\ \boxed{\bar{b}_{im1}^{u}\ \cdots\ \bar{b}_{imn_{im}^{bu}}^{u}} \\ & \boxed{c_{i1}\ \cdots\ c_{in_i^c}}\ \big]^T, \quad (N \text{ blocks})\end{aligned} \tag{3.12}$$

where $M = p + m + 1$, we obtain a pseudo-linear regression form

$$\hat{y}_i(t|\theta_i) = \varphi^T(t,\theta_i)\theta_i, \quad i = 1, \dots, p. \tag{3.13}$$

Remark 3.3. Note that there is an important relation between the framed parameter blocks in (3.12) and the network structure. Each arc in the digraph corresponds to a transfer function from input u_j or y_j to output y_i. The parameters of the transfer function are given in the block with parameters $b^u_{ij\cdot}$ or $b^y_{ij\cdot}$ together with $a_{i\cdot}$. For brevity, we will later (in (3.16) and (6.6)) denote these parameter blocks by $\mathbf{w}_k^{[c]}$ or $\mathbf{w}_k$ with a numbering k (see Figure 3.3 for an example).

3.3 Heterogeneous datasets

This section starts dealing with heterogeneous datasets, which will be integrated with additional constraints to guarantee Assumption 3.2. To avoid the lengthy index notations in (3.11) and (3.12), which are unnecessary in later discussions, we encapsulate them in new symbols.

3.3.1 Regression forms of multiple datasets

Consider the regression problems (3.13) in network reconstruction, where the p problems can be treated independently. Therefore, without loss of generality, it is assumed in later sections that we are dealing with the i-th output variable y_i by default. For simplicity, we introduce the following notations

$$\mathbf{y}^{[c]} \triangleq \begin{bmatrix} y_i(t_1|\theta_i) \\ \vdots \\ y_i(t_{N_c}|\theta_i) \end{bmatrix}, \quad \mathbf{A}^{[c]}(\mathbf{w}^{[c]}) \triangleq \begin{bmatrix} \varphi^T(t_1,\theta_i) \\ \vdots \\ \varphi^T(t_{N_c},\theta_i) \end{bmatrix}, \tag{3.14}$$

where $\mathbf{w}^{[c]} \triangleq \theta_i$; y_i, ϕ and θ_i correspond to the c-th experiment; and (3.13) is evaluated at $\{t_1,\dots,t_{N_c}\}$. We then have the following expression

$$\mathbf{y}^{[c]} = \mathbf{A}^{[c]}(\mathbf{w}^{[c]})\,\mathbf{w}^{[c]} + \boldsymbol{\xi}^{[c]}, \quad c = 1,\dots,C, \tag{3.15}$$

where

$$\begin{aligned} \mathbf{A}^{[c]} &\triangleq \begin{bmatrix} \mathbf{A}^{[c]}_{:,1} & \mathbf{A}^{[c]}_{:,2} & \cdots & \mathbf{A}^{[c]}_{:,M} \end{bmatrix}, \\ \mathbf{w}^{[c]} &\triangleq \Big[(\mathbf{w}^{[c]}_1)^T \quad \cdots \quad (\mathbf{w}^{[c]}_i)^T \quad \cdots \quad (\mathbf{w}^{[c]}_p)^T, \\ &\qquad (\mathbf{w}^{[c]}_{p+1})^T \quad \cdots \quad (\mathbf{w}^{[c]}_{p+i})^T \quad \cdots \quad (\mathbf{w}^{[c]}_{p+m})^T, \\ &\qquad (\mathbf{w}^{[c]}_M)^T \Big]^T \\ \boldsymbol{\xi}^{[c]} &\triangleq \begin{bmatrix} \xi^{[c]}(t_1) & \xi^{[c]}(t_2) & \cdots & \xi^{[c]}(t_{N_c}) \end{bmatrix}, \end{aligned} \tag{3.16}$$

$\mathbf{w}^{[c]}$ is partitioned into M blocks as illustrated in (3.12), $\mathbf{A}^{[c]}_{:,j}$ ($j = 1,\dots,M$) denotes the blocks of $\mathbf{A}^{[c]}$ that is partitioned correspondingly as $\mathbf{w}^{[c]}$, and $\boldsymbol{\xi}^{[c]}$ denotes the prediction error, which represents the part of the output $\mathbf{y}^{[c]}$ that cannot be predicted from past data using the chosen model classes.

Letting

$$\mathbf{w}_k \triangleq \left[\begin{array}{c} \mathbf{w}_k^{[1]} \\ \hline \vdots \\ \hline \mathbf{w}_k^{[C]} \end{array}\right], \quad \mathbf{w} = \left[\begin{array}{c} \mathbf{w}_1 \\ \hline \vdots \\ \hline \mathbf{w}_N \end{array}\right], \tag{3.17}$$

we integrate all datasets by stacking (6.3) for each dataset and rearranging blocks of matrices, yielding (6.5), and, for simplicity, use $\mathbf{y} = \mathbf{A}(\mathbf{w})\mathbf{w} + \boldsymbol{\xi}$ to denote (3.18b).

$$\begin{bmatrix} \mathbf{y}^{[1]} \\ \vdots \\ \mathbf{y}^{[C]} \end{bmatrix} = \underbrace{\left[\begin{array}{c|c|c} \mathbf{A}_{:,1}^{[1]} \ \dots \ \mathbf{A}_{:,M}^{[1]} & & \\ & \ddots & \\ & & \mathbf{A}_{:,1}^{[C]} \ \dots \ \mathbf{A}_{:,M}^{[C]} \end{array}\right]}_{C \ \textbf{Blocks}} \left[\begin{array}{c} \mathbf{w}^{[1]} \\ \hline \vdots \\ \hline \mathbf{w}^{[C]} \end{array}\right] + \left[\begin{array}{c} \boldsymbol{\xi}^{[1]} \\ \hline \vdots \\ \hline \boldsymbol{\xi}^{[C]} \end{array}\right] \tag{3.18a}$$

$$= \underbrace{\left[\begin{array}{ccc|c|ccc} \mathbf{A}_{:,1}^{[1]} & & & & \mathbf{A}_{:,M}^{[1]} & & \\ & \ddots & & \dots & & \ddots & \\ & & \mathbf{A}_{:,1}^{[C]} & & & & \mathbf{A}_{:,M}^{[C]} \end{array}\right]}_{M \ \textbf{Blocks}} \left[\begin{array}{c} \mathbf{w}_1 \\ \hline \vdots \\ \hline \mathbf{w}_M \end{array}\right] + \left[\begin{array}{c} \boldsymbol{\xi}^{[1]} \\ \hline \vdots \\ \hline \boldsymbol{\xi}^{[C]} \end{array}\right] \tag{3.18b}$$

Remark 3.4. When experiments are perfectly repeated, i.e. the homogeneous case (see Section 3.1.2), we have the ideal case $\mathbf{w}^{[1]} = \cdots = \mathbf{w}^{[C]} \equiv \mathbf{w}$. A simple linear regression form is formulated for identification by concatenation: we have a simple setup as follows

$$\begin{bmatrix} \mathbf{y}^{[1]} \\ \vdots \\ \mathbf{y}^{[C]} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{[1]}(\mathbf{w}) \\ \vdots \\ \mathbf{A}^{[C]}(\mathbf{w}) \end{bmatrix} \mathbf{w} + \begin{bmatrix} \boldsymbol{\xi}^{[1]} \\ \vdots \\ \boldsymbol{\xi}^{[C]} \end{bmatrix}. \tag{3.19}$$

3.3.2 Simultaneous sparsity regularization

Now we consider the two essential requirements for network reconstruction from heterogeneous datasets: 1) sparse networks is acquired in the presence of noise; 2) $\mathbf{w}^{[c]}$ is required to give the same network topology for all c, i.e. the inference results $(\hat{Q}, \hat{P}, \hat{H})^{[c]}$ satisfy $\mathcal{G}((\hat{Q}, \hat{P}, \hat{H})^{[c]}) \equiv \mathcal{G}^0, \forall c$ (see Section 3.1.2).

Let us introduce a term of group sparsity based on $\mathbf{w}$,

$$\mathbf{w}^S := [\|\mathbf{w}_1\|_2, \ \cdots, \ \|\mathbf{w}_M\|_2]^T, \tag{3.20}$$

where $\|\cdot\|_2$ denotes the l_2-norm of vectors and M is the number of groups/blocks in (3.16). The dimensions of each block $\mathbf{w}_k^{[c]}$ and $\mathbf{w}_k$ in (3.17) are saved in two vectors ρ^E and ρ^S,

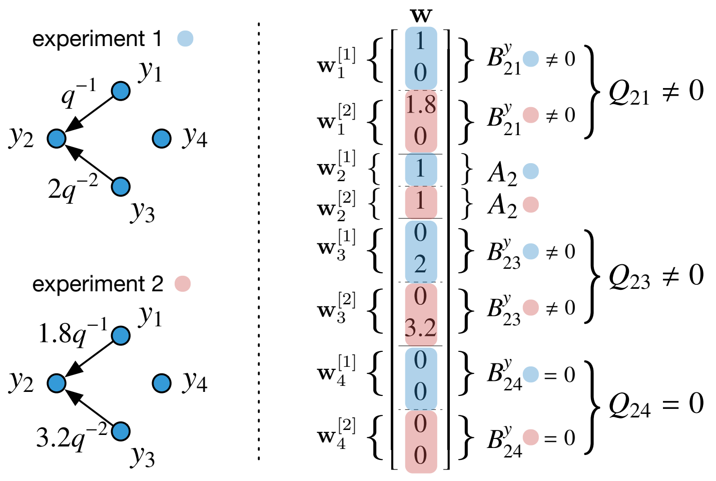


Figure 3.3. An example of $\mathbf{w}$ in the setup of multiple experiments.

respectively

$$\begin{aligned} &\rho \triangleq [n_{i1}^{by}, \dots, n_i^a, \dots, n_{ip}^{by}, n_{i1}^{bu}, \dots, n_{ip}^{bu}, n_i^c]^T, \\ &\rho^E := \rho \otimes \mathbf{1}_C, \qquad \rho^S := C\rho, \end{aligned} \tag{3.21}$$

where the elements of ρ are defined with respect to (3.12), $\mathbf{1}_C$ is a C-dimensional column vector of 1's, and the index i in ρ indicates the regression problem for y_i, as assumed by default.

Group sparsity is demanded to guarantee networks sparsity and that the network structure is consistent over replica (i.e. the interconnection structure determined by the $\mathbf{w}^{[c]}$'s are identical). The sparsity is imposed on each large group $\mathbf{w}_k$, $k = 1, \dots, M$, and the penalty term is $\lambda\|\mathbf{w}^S\|_0$, where $\lambda \in \mathbb{R}^+$. The mechanism on how the group sparsity functions is described as follows.

Recall that the setup (6.5) allows the system parameters to be different in values for different c's. Note that each small block $\mathbf{w}_k^{[c]}$ corresponds to an arc in the underlying digraph of the dynamical system in the c-th experiment (e.g., see Figure 3.3). The $\|\mathbf{w}_k\|_2$ chosen to be zero yields that all $\mathbf{w}_k^{[c]}, c = 1, \dots, C$ are equal to zeros. It implies that the arc corresponding to $\mathbf{w}_k^{[c]}$ does not exist in dynamic networks for any c. In addition, thanks to the effect of noise, it is nearly guaranteed that $\mathbf{w}_k^{[c]}$ is not identical to zero for almost all c if $\|\mathbf{w}_k\|_2 \neq 0$. This is how the group sparsity (defined via $\mathbf{w}^S$) guarantees that the resultant networks of different datasets share the same topology. Moreover, when a classical least squares objective is augmented with a penalty term of $\lambda\|\mathbf{w}^S\|_0$, the optimal solution favors zeros of $\mathbf{w}_k, k = 1, \dots, M$, which guarantees the sparsity of network structures.

In summary, to perform network reconstruction from heterogeneous datasets, we solve the following optimization problem

$$\underset{\mathbf{w}}{\text{minimize}}\ \|\mathbf{y} - \mathbf{A}(\mathbf{w})\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}^S\|_0. \tag{3.22}$$

Remark 3.5. In programming, one may need to find all elements of $\mathbf{w}$ that correspond to the k^E-th small group of $\mathbf{w}$ (i.e. the k^E-th vector $\mathbf{w}_k^{[c]}$ in $\mathbf{w}$) or the k^S-th large group of $\mathbf{w}$ (i.e. the vector $\mathbf{w}_{k^S}$). Here are the formulas:

- $k = (C\sum_{j=1}^{\lceil k^E/C\rceil-1}\rho_j)\cdot\mathbf{1}+[((k^E-1) \bmod C)\rho_{\lceil k^E/C\rceil}+1]:1:((k^E-1) \bmod C+1)\rho_{\lceil k^E/C\rceil}$
- $k = \left(C\sum_{j=1}^{k^S-1}\rho_j\right)\cdot\mathbf{1}+1:1:C\rho_{k^S}$

where ρ is given in (3.21), $\mathbf{1}$ is a row vector of 1's of the matched dimension, $m:1:n$ $(m, n \in \mathbb{N}_+)$ denotes a row vector $[m, m+1, \dots, n]$, and $\lceil x \rceil$ denotes the smallest natural number that is larger than x.

3.4 Methods for group sparsity

The section presents three methods to acquire sparse network structures, i.e. the estimation of $\mathbf{w}$ is group-wise sparse with the fixed partition given in (3.17). Due to the limitations of these methods (the regression (3.13) has to be linear), we are only able to handle the ARX case, i.e. (3.22) is a zero-norm regularized linear least square problem. Section 3.4.1 introduces the classical l_1 approach, which uses the best convex approximation of zero norm. Section 3.4.2 uses sparse Bayesian learning, proposed in Tipping (2001), which is further extended to deal with the case with different group sizes. And a sampling method is used in Section 3.4.3, which is based on the work of model selection in Kuo and Mallick (1998) and is extended to handle a more general case.

3.4.1 Classical l_1 methods

Convex approximation

As addressed in Section 3.2.3, choosing ARX to parametrize network models results in a linear regression, in which $\mathbf{A}$ does not depend on $\mathbf{w}$ in (6.5). The treatment of classical group LASSO yields

$$\underset{\mathbf{w}}{\text{minimize}}\ \|\mathbf{y}-\mathbf{A}\mathbf{w}\|_2^2+\lambda\|\mathbf{w}^S\|_1, \tag{3.23}$$

where

$$\|\mathbf{w}^S\|_1=\sum_{i=1}^{N}\sqrt{\rho_i^S}\|\mathbf{w}_i\|_2, \tag{3.24}$$

and $\lambda \in \mathbb{N}_+$. This is a convex optimization and has been soundly studied (e.g., Yuan and Lin (2006)).

To achieve a better approximation of the l_0-norm, alternatively one may use *iterative reweighted* l_1/l_2 *methods* (e.g. see Candes et al. (2008); Chartrand and Yin (2008)). Applying to group sparsity, both methods turn to a similar scheme (differing in the usage of $\|\cdot\|_2$ or

$\|\cdot\|_2^2$ for blocks of $\mathbf{w}$ in (3.25) and (3.26)). Here we present the solution using the l_1 method.

$$\mathbf{w}^{(k+1)} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{A}\mathbf{w}\|_2^2 + \lambda \sum_{i=1}^{N} \nu_i^{(k)} \sqrt{\rho_i^S} \|\mathbf{w}_i\|_2, \tag{3.25}$$

where

$$\nu_i^{(k)} = \left[\|\mathbf{w}_i^{(k)}\|_2 + \epsilon^{(k)}\right]^{-1} \tag{3.26}$$

and k is the index of iterations. In regard to the selection of ϵ, $\{\epsilon^{(k)}\}_{k=1,2,\dots}$ should be a sequence converging to zero, as addressed in Chartrand and Yin (2008) based on the *Unique Representation Property*. It suggests in Chartrand and Yin (2008) a fairly simple update rule of ϵ, i.e. $\epsilon^{(k)} \in (0,1)$ is reduced by a factor of 10 until reaching a minimum of 10^{-8} (the factor and lower bound could be tuned specifically). One may also adopt an adaptive rule of ϵ given in Candes et al. (2008).

ADMM for large-scale problems

To solve the convex optimization in Section 3.4.1, for example, *CVX* for MATLAB could be an easy solution. However, the computation time could be enormous for large-dimension problems. This section presents algorithms using *proximal methods* and *ADMM* (Parikh and Boyd (2013)) to handle large-dimension network reconstruction.

Let us first consider (3.23), which is rewritten as

$$\underset{\mathbf{w}}{\text{minimize}}\ f(\mathbf{w}) + g(\mathbf{w}), \tag{3.27}$$

where $f(\mathbf{w}) \triangleq (1/2)\|\mathbf{y} - \mathbf{A}\mathbf{w}\|_2^2$, $g(\mathbf{w}) \triangleq \lambda\|\mathbf{w}^S\|_1$, λ is twice larger than the value in (3.24). Given $\nabla f(\mathbf{w}) = \mathbf{A}^T(\mathbf{A}\mathbf{w} - \mathbf{y})$, the *proximal gradient method* is to update $\mathbf{w}$ by $\mathbf{w}^{(k+1)} = \mathbf{prox}_{\gamma g}(\mathbf{w}^{(k)} - \gamma\nabla f(\mathbf{w}^{(k)}))$, $\gamma \in \mathbb{R}_+$, where k is the iteration index and $\mathbf{prox}_f(\cdot)$ denotes the standard *proximal operator* of function f (see Boyd et al. (2011); Parikh and Boyd (2013)). It is easy to see that $g(\mathbf{w}) = \sum_{i=1}^N g_i(\mathbf{w}_i)$, where $g_i(\mathbf{w}_i) := \lambda\sqrt{\rho_i^S}\|\mathbf{w}_i\|_2$. Firstly we partition the variable $\mathbf{v}$ of $\mathbf{prox}_{\gamma g}(\mathbf{v})$ in the same way as $\mathbf{w}$ in terms of $\mathbf{w}_i, i = 1, \dots, N$, i.e. $\mathbf{v} = [\mathbf{v}_1^T, \dots, \mathbf{v}_N^T]^T$. Then we calculate the proximal operator $\mathbf{prox}_{\gamma g_i}(\mathbf{v}_i)$, which equals

$$\mathbf{prox}_{\gamma g_i}(\mathbf{v}_i) = \left(1 - \gamma\lambda\sqrt{\rho_i^S}/\|\mathbf{v}_i\|_2\right)_+ \mathbf{v}_i, \tag{3.28}$$

where $(\cdot)_+$ replaces each negative elements with 0. It follows that

$$\mathbf{prox}_{\gamma g}(\mathbf{v}) = \left[\left(\mathbf{prox}_{\gamma g_1}(\mathbf{v}_1)\right)^T \ \cdots \ \left(\mathbf{prox}_{\gamma g_N}(\mathbf{v}_N)\right)^T\right]^T. \tag{3.29}$$

The value of γ needs to be selected appropriately so as to guarantee the convergence. One simple solution is using line search methods, e.g., see Section 4.2 in Parikh and Boyd (2013).

Provided with the above calculations, it is straightforward to implement the *(accelerated) proximal gradient method* (see (Parikh and Boyd, 2013, chap. 4.3)). To implement ADMM, the proximal operator of $f(\mathbf{w})$ needs to be calculated,

$$\mathbf{prox}_{\gamma f}(\mathbf{v}) = (I + \gamma \mathbf{A}^T\mathbf{A})^{-1}(\gamma \mathbf{A}^T\mathbf{y} + \mathbf{v}). \tag{3.30}$$

Given $\mathbf{prox}_{\gamma g}(\mathbf{v})$ as (3.28) and (3.29), the ADMM method is presented in Algorithm 6.

Algorithm 2 ADMM method

```
1:  Precompute $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{y}$
2:  given an initial value $\mathbf{w}^0, \mathbf{z}^0, \mathbf{u}^0$, $\gamma^0 = 1$, and $\beta = 1/2$
3:  repeat
4:      $\gamma \leftarrow \gamma^{(k)}$
5:      repeat
6:          $\hat{\mathbf{w}} \leftarrow \mathbf{prox}_{\gamma f}(\mathbf{z}^{(k)} - \mathbf{u}^{(k)})$ using (6.14)
7:          break if $f(\hat{\mathbf{w}}) \leq f(\mathbf{w}^{(k)}) + \nabla f(\mathbf{w}^{(k)})^T(\hat{\mathbf{w}} - \mathbf{w}^{(k)}) + (1/2\gamma)\|\hat{\mathbf{w}} - \mathbf{w}^{(k)}\|_2^2$
8:          $\gamma \leftarrow \beta\gamma$
9:      until ;
10:     $\mathbf{w}^{(k+1)} \leftarrow \hat{\mathbf{w}}$, $\gamma^{(k+1)} \leftarrow \gamma$
11:     Compute $\mathbf{prox}_{\gamma g_i}(\mathbf{w}_i^{(k+1)} + \mathbf{u}_i^{(k)})$ by (3.28) for $i = 1, ..., N$
12:     $\mathbf{z}^{(k+1)} \leftarrow \mathbf{prox}_{\gamma g}(\mathbf{w}^{(k+1)} + \mathbf{u}^{(k)})$ using (3.29)
13:     $\mathbf{u}^{(k+1)} \leftarrow \mathbf{u}^{(k)} + \mathbf{w}^{(k+1)} - \mathbf{z}^{(k+1)}$
14: until any standard stopping criteria
```

To use this algorithm for the iterative reweighted l_1 method (3.25), we only need to modify (3.28), which now should be

$$\mathbf{prox}_{\gamma g_i}(\mathbf{v}_i) = \left(1 - \gamma\lambda\nu_i\sqrt{\rho_i^S}/\|\mathbf{v}_i\|_2\right)_+ \mathbf{v}_i. \tag{3.31}$$

In each "outer" loop indicated by (3.25), we update ν_i by (3.26) and implement ADMM as Algorithm 6 to solve (3.25).

3.4.2 Sparse Bayesian learning

This section uses an empirical Bayesian algorithm to achieve group sparsity, which is proposed in Tipping (2001). Consider the following model

$$\mathbf{y} = \mathbf{A}\mathbf{w} + \boldsymbol{\xi}, \quad \boldsymbol{\xi} \sim \mathcal{N}(0, \Sigma), \tag{3.32}$$

where $\Sigma = \operatorname{diag}(\sigma_1^2 I_1, \ldots, \sigma_C^2 I_C)$ with $\sigma_c \in \mathbb{R}_+$ and identity matrix I_c is of the dimension M_c indicated in (6.3) for $i = 1, \ldots, C$; $\mathbf{y}$ is of dimension $M_\mathbf{y}$; and $\mathbf{w}$ is of dimension $M_\mathbf{w}$ (easy to see $M_\mathbf{y} = \sum_{c=1}^{C} N_c$ and $N_\mathbf{w} = \|\rho^S\|_1$ from (6.3) and (6.6)). In general, it might be too strict to approximately assume Σ has the simple form $\Sigma = \sigma^2 I$ used in most SBL papers (e.g., Tipping (2001); Wipf and Nagarajan (2010); Wipf and Rao (2004, 2007)). The reason is the heterogeneity of datasets, which allows the noise variances in different experiments to be different. When introducing sparse Bayesian learning (SBL), this section mainly extends it in two ways: dealing with different group sizes and handling general diagonal variance matrices.

Model prior formulation

Suppose the *automatic relevance determination* (ARD) prior Tipping (2001); Wipf and Rao (2004) incorporated by SBL is given as

$$p(\mathbf{w};\Gamma) = \frac{1}{(2\pi)^{M_{\mathbf{w}}/2}|\Gamma|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{w}^T\Gamma^{-1}\mathbf{w}\right). \tag{3.33}$$

In regard to hyperparameter Γ, we will enforce a specific structure empirically to impose group sparsity, as proposed in Wipf and Rao (2007). For each $\mathbf{w}_j$, $j = 1, \dots, M$,

$$p(\mathbf{w}_j;\gamma_j) = \mathcal{N}(0, \gamma_j I_j), \tag{3.34a}$$

$$p(\mathbf{w};\boldsymbol{\gamma}) = \prod_{j=1}^{M} p(\mathbf{w}_j;\gamma_j), \tag{3.34b}$$

where $\gamma_j \in \mathbb{R}_+$, I_j is a $CM_j \times CM_j$ identity matrix,

$$M_j = \begin{cases} n_{ij}^{by} & \text{if } 1 \le j \le p \text{ and } j \ne i, \\ n_i^a & \text{if } j = i, \\ n_{ij}^{bu} & \text{if } p+1 \le j \le p+m = M, \end{cases} \tag{3.35}$$

the index i in (3.35) indicates that we are dealing with the i-th output (see (3.12), (3.14)), and $\boldsymbol{\gamma} \triangleq [\gamma_1, \dots, \gamma_M]^T$. Therefore, Γ is constructed as

$$\Gamma \triangleq \operatorname{diag}(\gamma_1 I_1, \cdots, \gamma_M I_M). \tag{3.36}$$

For simplicity, we will interchangeably use two notations for the prior $p(\mathbf{w};\Gamma) \triangleq p(\mathbf{w};\boldsymbol{\gamma})$.

Parameter estimation

The likelihood function of (3.32) is Gaussian,

$$p(\mathbf{y} \mid \mathbf{w};\Sigma) = (2\pi)^{-\frac{M_{\mathbf{y}}}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(\|\mathbf{y} - \mathbf{A}\mathbf{w}\|^2_{\Sigma^{-1}}\right). \tag{3.37}$$

The ARD prior incorporated by SBL is (3.33). The hyperparameters $\boldsymbol{\gamma}$ (along with the error variance Σ if necessary) can be estimated from the data by marginalizing over the weights $\mathbf{w}$ and then perform *maximum likelihood* optimization. The marginalized probabilistic density function is

$$\begin{aligned} p(\mathbf{y};\boldsymbol{\gamma},\Sigma) &= \int p(\mathbf{y} \mid \mathbf{w};\Sigma)p(\mathbf{w};\boldsymbol{\gamma})\mathrm{d}\mathbf{w} \\ &= \frac{1}{(2\pi)^{M_{\mathbf{y}}/2}|\Sigma_{\mathbf{y}}|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{y}^T\Sigma_{\mathbf{y}}^{-1}\mathbf{y}\right), \end{aligned} \tag{3.38}$$

where $\Sigma_{\mathbf{y}} \triangleq \Sigma + \mathbf{A}\Gamma\mathbf{A}^T$. The procedure is referred to as *evidence maximization* or *type-II maximum likelihood.* With fixed values of the hyperparameters, the posterior density is Gaussian, i.e.

$$p(\mathbf{w} \mid \mathbf{y};\boldsymbol{\gamma},\Sigma) = \mathcal{N}(\boldsymbol{\mu},\Sigma_{\mathbf{w}}) \tag{3.39}$$

with $\boldsymbol{\mu} = \Sigma_{\mathbf{w}}\mathbf{A}^T\Sigma^{-1}\mathbf{y}$ and $\Sigma_{\mathbf{w}} = \left(\mathbf{A}^T\Sigma^{-1}\mathbf{A} + \Gamma^{-1}\right)^{-1}$. Once we have $\boldsymbol{\gamma}$ and Σ estimated via type-II maximum likelihood, we can choose as our weights the $\hat{\mathbf{w}}$ satisfying

$$\hat{\mathbf{w}} = \boldsymbol{\mu} = \left(\mathbf{A}^T\Sigma_{\mathrm{ML}}^{-1}\mathbf{A} + \Gamma_{\mathrm{ML}}^{-1}\right)^{-1}\mathbf{A}^T\Sigma_{\mathrm{ML}}^{-1}\mathbf{y}. \tag{3.40}$$

Algorithms

There are several ways to find Γ_{ML} and Σ_{ML}: *expectation maximization* (EM) in Wipf and Rao (2004), fixed-point methods in Tipping (2001), and *difference of convex program* (DCP) in Pan et al. (2015). We adopt the EM approach here and the others can be similarly derived.

The EM method proceeds by treating the weights $\mathbf{w}$ as hidden variables and then maximizing

$$\mathbb{E}_{\mathbf{w}|\mathbf{y};\boldsymbol{\gamma},\Sigma}\left[p(\mathbf{y},\mathbf{w};\boldsymbol{\gamma},\Sigma)\right],$$

where $p(\mathbf{y},\mathbf{w};\boldsymbol{\gamma},\Sigma) = p(\mathbf{y}|\mathbf{w};\Sigma)p(\mathbf{w};\boldsymbol{\gamma})$ is the likelihood of the complete data $\{\mathbf{w},\mathbf{y}\}$. The whole EM algorithm for the extended SBL is summarized as follows: given the estimates $\boldsymbol{\gamma}^{(k)}$ and $\Sigma^{(k)}$, at the $(k+1)$-th iterate,

- E-step: $\mathbb{E}_{\mathbf{w}|\mathbf{y};\boldsymbol{\gamma}^{(k)},\Sigma^{(k)}}(\mathbf{w}_i^2) = (\Sigma_{\mathbf{w}})_{i,i} + \boldsymbol{\mu}_i^2$.
- M-step:

$$\begin{aligned} \boldsymbol{\gamma}^{(k+1)} &= \underset{\boldsymbol{\gamma}\geq 0}{\operatorname{argmax}}\ \mathbb{E}_{\mathbf{w}|\mathbf{y};\boldsymbol{\gamma}^{(k)},\Sigma^{(k)}}\left[p(\mathbf{y},\mathbf{w};\boldsymbol{\gamma},\Sigma^{(k)})\right] \\ \Rightarrow \quad \gamma_j^{(k+1)} &= \tfrac{1}{CM_j}\textstyle\sum_{i\in\Lambda_j}\left[\boldsymbol{\mu}_i^2 + (\Sigma_{\mathbf{w}})_{i,i}\right], \\ \Sigma^{(k+1)} &= \underset{\Sigma>0}{\arg\max}\ \mathbb{E}_{\mathbf{w}|\mathbf{y};\boldsymbol{\gamma}^{(k)},\Sigma^{(k)}}\left[p(\mathbf{y},\mathbf{w};\boldsymbol{\gamma}^{(k)},\Sigma)\right] \\ \Rightarrow \quad (\sigma_c^2)^{(k+1)} &= \tfrac{1}{N_{\mathbf{y}}}\{\|\mathbf{y}^{[c]} - \mathbf{A}^{[c]}\boldsymbol{\mu}\|^2 + (\sigma_c^2)^{(k)}\textstyle\sum_{i=1}^{M_{\mathbf{w}}}\left[1 - (\boldsymbol{\gamma}_i^{(k)})^{-1}(\Sigma_{\mathbf{w}})_{i,i}\right]\}, \end{aligned}$$

for $j = 1,\ldots,M$; $i = 1,\ldots,M_{\mathbf{w}}$; $c = 1,\ldots,C$ and $\Sigma^{(k+1)} = \mathrm{diag}((\sigma_1^2)^{(k+1)}I_1,\ldots,(\sigma_C^2)^{(k+1)}I_C)$, where $\boldsymbol{\mu}$ and $\Sigma_{\mathbf{w}}$ are calculated via (3.39) provided with $\boldsymbol{\gamma}^{(k)}$ and $\Sigma^{(k)}$, M_j is given in (3.35) and Λ_j denotes the row (column) indexes associated with γ_j in Γ.

3.4.3 Sampling methods

The sampling method for sparsity is based on the work of model selection in Kuo and Mallick (1998), which is further extended to deal with group sparsity with different sizes. Consider the same model (3.32) in Section 3.4.2 and the likelihood is given by (3.37). We choose the same Gaussian prior as (3.33), where Γ can be assumed to be either simply $\Gamma = \gamma I$ or diagonal, i.e. $\Gamma = \text{diag}(\gamma_1 I_1, \ldots, \gamma_M I_M)$. Letting $\boldsymbol{\gamma} \triangleq (\gamma_1, \ldots, \gamma_M)$[1], for brevity, we will use Γ and $\boldsymbol{\gamma}$ interchangeably. Furthermore, we adopt the inverse Gamma distribution as the hyperpriors for σ^2 and $\boldsymbol{\gamma}$ (e.g., see Berger (1993)), denoted by $\sigma^2 \sim \mathcal{G}^{-1}(a, b)$ and $\boldsymbol{\gamma} \sim \prod_{i=1}^{N} \mathcal{G}^{-1}(c, d)$, where a, b, c, d choose small values, e.g., $a = b = c = d = 10^{-4}$. The key to acquire sparsity is to sample the indicator variable $\mathbf{s} \triangleq [s_1, \ldots, s_N]$, which is introduced as follows

$$\mathbf{w} = \text{diag}(s_1 I_1, \ldots, s_M I_M)\boldsymbol{\vartheta} \triangleq S\boldsymbol{\vartheta}, \tag{3.41}$$

where $s_i \in \{0, 1\}$, I_j $(j = 1, \ldots, M)$ is given as (3.35), and $\boldsymbol{\vartheta} \in \mathbb{R}^{M_{\mathbf{w}}}$. We choose the Bernoulli distribution for the prior of $\mathbf{s}$, denoted by $\mathbf{s} \sim \prod_{i=1}^{M} \mathcal{B}(1, p_i)$ $(p_i \in (0, 1)$, e.g., $p_i = 1/2)$. Being different from Kuo and Mallick (1998), we will not sample $\mathbf{s}$ and $\boldsymbol{\vartheta}$ at the same time, due to the considerable cost on sampling $\boldsymbol{\vartheta}$. Instead we sample $\mathbf{s}$ and determine the sparse structure first, and then run a linear regression to calculate $\mathbf{w}$. The onus is to sample $\mathbf{s}, \Sigma$ and $\boldsymbol{\gamma}$ from $p(\mathbf{s}, \boldsymbol{\gamma}, \Sigma \mid \mathbf{y})$. We marginalize the posterior $p(\mathbf{s}, \boldsymbol{\vartheta}, \boldsymbol{\gamma}, \Sigma \mid \mathbf{y})$ over $\boldsymbol{\vartheta}$,

$$\begin{aligned} p(\mathbf{s}, \boldsymbol{\gamma}, \Sigma \mid \mathbf{y}) &\propto p(\mathbf{s})p(\boldsymbol{\gamma})p(\Sigma)p(\mathbf{y} \mid \mathbf{s}, \Sigma, \boldsymbol{\gamma}), \\ &\propto p(\mathbf{s})p(\boldsymbol{\gamma})p(\Sigma) \int p(\mathbf{y} \mid \mathbf{w}, \Sigma)p(\boldsymbol{\vartheta} \mid \boldsymbol{\gamma})\mathrm{d}\boldsymbol{\vartheta}. \end{aligned} \tag{3.42}$$

Substituting the prior of $\boldsymbol{\vartheta}$ and completing squares, it yields

$$p(\mathbf{y} \mid \mathbf{s}, \Sigma, \boldsymbol{\gamma}) = (2\pi)^{-\frac{M_{\mathbf{y}}}{2}} \left|\Sigma_{\mathbf{y}}\right|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\|\mathbf{y}\|^2_{\Sigma_{\mathbf{y}}^{-1}}\right), \tag{3.43}$$

where

$$\Sigma_{\mathbf{y}} = \Sigma + \mathbf{A}S\Gamma S\mathbf{A}^T. \tag{3.44}$$

We use the *Metropolis-Hastings* algorithm within the *systematic-scan Gibbs sampler* (e.g., see Liu (2008)) to draw samples of $\mathbf{s}, \boldsymbol{\gamma}$ and Σ from their joint distribution $p(\mathbf{s}, \boldsymbol{\gamma}, \Sigma \mid \mathbf{y})$. Suppose that the k-th samples $\mathbf{s}^{(k)}, \boldsymbol{\gamma}^{(k)}$ and $\Sigma^{(k)}$ have been available. At the $k+1$ iteration,

- draw $\mathbf{s}^{(k+1)}$ from the conditional distribution $p(\mathbf{s} \mid \mathbf{y}, \boldsymbol{\gamma}^{(k)}, \Sigma^{(k)})$,
- draw $\boldsymbol{\gamma}^{(k+1)}$ from $p(\boldsymbol{\gamma} \mid \mathbf{y}, \mathbf{s}^{(k+1)}, \Sigma^{(k)})$, and
- draw $\Sigma^{(k+1)}$ from $p(\Sigma \mid \mathbf{y}, \mathbf{s}^{(k+1)}, \boldsymbol{\gamma}^{(k+1)})$.

[1] If using $\Gamma = \gamma I$, $\boldsymbol{\gamma}$ becomes a simple scalar. The difference from the SBL is that the position of Γ in sparsity pursuit has been largely weaken.

In each sampling step, we use the *Metropolis-Hastings* algorithm. Consider drawing samples from $p(\mathbf{s} \mid \mathbf{y}, \boldsymbol{\gamma}, \Sigma)$, in which the key is to choose a proposal distribution $T(\mathbf{s}^{(k)}, \hat{\mathbf{s}})$, where $\mathbf{s}^{(k)}$ is a given sample and $\hat{\mathbf{s}}$ is a new configuration. Here we use a simple scheme that gives a symmetric proposal distribution (i.e. $T(\mathbf{s}^{(k)}, \hat{\mathbf{s}}) = T(\hat{\mathbf{s}}, \mathbf{s}^{(k)})$). We randomly pick an element of $\mathbf{s}^{(k)}$ and flip it, i.e. draw $\hat{i}$ from the uniform distribution on $\{1, \dots, N\}$ and set, for $i = 1, \dots, N$,

$$\hat{s}_i = \begin{cases} s_i^{(k)}, & \text{if } i \neq \hat{i}, \\ 1 - s_i^{(k)}, & \text{if } i = \hat{i}. \end{cases} \tag{3.45}$$

The Metropolis-Hastings algorithm for $p(\mathbf{s} \mid \mathbf{y}, \boldsymbol{\gamma}, \Sigma)$ is given as below: given $\boldsymbol{\gamma}^{(k)}, \Sigma^{(k)}$ and $\mathbf{s}^{(k)}$,

- draw $\hat{\mathbf{s}}$ from the proposal distribution $T(\mathbf{s}^{(k)}, \hat{\mathbf{s}})$ using the above scheme;
- draw $U \sim \text{Uniform}[0, 1]$ and update

$$\mathbf{s}^{(k+1)} = \begin{cases} \hat{\mathbf{s}}, & \text{if } U \leq r(\mathbf{s}^{(k)}, \hat{\mathbf{s}}) \\ \mathbf{s}^{(k)}, & \text{otherwise.} \end{cases}$$

  where

$$r(\mathbf{s}^{(k)}, \hat{\mathbf{s}}) = \min\left\{1, \frac{p(\hat{\mathbf{s}}, \boldsymbol{\gamma}^{(k)}, \Sigma^{(k)} \mid \mathbf{y}) T(\hat{\mathbf{s}}, \mathbf{s}^{(k)})}{p(\mathbf{s}^{(k)}, \boldsymbol{\gamma}^{(k)}, \Sigma^{(k)} \mid \mathbf{y}) T(\mathbf{s}^{(k)}, \hat{\mathbf{s}})}\right\}.$$

In regard to the Metropolis-Hastings algorithms for $\boldsymbol{\gamma}$ and Σ, we use random walk sampling $\hat{\boldsymbol{\gamma}} = \boldsymbol{\gamma}^{(k)} + \mathbf{u}, \hat{\Sigma} = \Sigma^{(k)} + \text{diag}(v_1 I_1, \dots, v_C I_C)$, where $\mathbf{u}$ and v_i are normally distributed with mean zero and fixed variances (which may need to be tuned to have sound convergence speeds and acceptance ratios), and the acceptance probabilities are given as, respectively,

$$\begin{aligned} r(\boldsymbol{\gamma}^{(k)}, \hat{\boldsymbol{\gamma}}) &= \min\left\{1, \frac{p(\mathbf{s}^{(k+1)}, \hat{\boldsymbol{\gamma}}, \Sigma^{(k)} \mid \mathbf{y})}{p(\mathbf{s}^{(k+1)}, \boldsymbol{\gamma}^{(k)}, \Sigma^{(k)} \mid \mathbf{y})}\right\}, \\ r(\Sigma^{(k)}, \hat{\Sigma}) &= \min\left\{1, \frac{p(\mathbf{s}^{(k+1)}, \boldsymbol{\gamma}^{(k+1)}, \hat{\Sigma} \mid \mathbf{y})}{p(\mathbf{s}^{(k+1)}, \boldsymbol{\gamma}^{(k+1)}, \Sigma^{(k)} \mid \mathbf{y})}\right\}. \end{aligned}$$

3.5 Numerical examples

3.5.1 ARX networks

The term "ARX network" refers to such a discrete-time DSF model (3.3) that each MISO transfer function (3.4) can be exactly written as an ARX model. This section mainly consists of two subjects: 1) model generation and simulation of "stable" ARX networks with random sparse network topology, 2) benchmark studies of the proposed inference methods.

Model generation of ARX networks is not straightforward due to the requirements of stability and sparsity. First we need to clarify these three words that quantifies or specifies the ARX networks our study: “random”, “sparse” and “stable”. The word “random” demands that both network structures and model parameters are randomly generated. The sparsity demands the network structures of ARX models being sparse; meanwhile we avoid such sparse structures that the network degenerates into a family of separate small networks, or the network has oversimplified structures, e.g. a tree with depth 2 (models generated by `drmodel.m` in MATLAB). Our simulation ensures that the generated networks are not acyclic, since the feedback loops are essential in physical systems but challenging to deal with in network reconstruction. The stability of ARX networks derives from the definition of network stability for DSFs (see (Yue, 2018, chap. 2.3)), which requests that each polynomial in $A(q)$ (our simulation also includes $B^y(q)$ and $B^u(q)$) is stable (i.e. all roots stay inside of the unit circle on the complex plane), and the resultant MIMO transfer function (from u to y) is stable. The difficulty on model generation is due to the latter requirement, since the DSF model with random sparse network topology may not be BIBO stable even if each entry (SISO transfer function) in Q and P has been chosen to be stable and even minimal-phase.

The idea to guarantee stable ARX networks is to repetitively apply the *small gain theorem* (e.g., see (Zhou and Doyle, 1998, p. 137)). First we generate a random sparse Boolean matrix Q^o, which specifies the network structure (to ease later discussions, we use the transpose of adjacency matrices), with zero diagonal and nonzero values of $Q^o_{k,k-1}$, e.g., Figure 3.4. The

$$Q^o = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

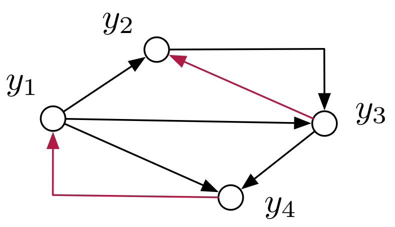


Figure 3.4. An example of the Boolean matrix and its digraph.

lower triangular part of Q^o specifies a block diagram (with unknown transfer functions) with only feedforward paths[2], and the upper triangular part adds feedback loops (marked in red in Figure 3.4). The key is to use the *small gain theorem* to tune the added feedback transfer functions one by one to guarantee BIBO stability. The transfer matrix $Q(q)$ with structure Q^o is created by generating random stable polynomials for nonzero entries in A, B^y and B^u, where nonzero entries of B^y are specified by Q^o, and then using (3.7). The last step is to tune the gain of each feedback transfer function, as specified by Q^o_U, using the *small gain theorem* sequentially. Here we will present an example to show the whole procedure. Suppose that the random structure for the 4-node network is specified by Q^o and $P^o = [1\ 0\ 0\ 0]^T$ (the Boolean matrix of P), and the block diagram is shown in Figure 3.5, where SISO transfer

[2]By default, we use the order $y_1 \to y_2 \to y_3 \to y_4$ as the forward path. It is certainly free to define any order as the default forward path.

functions ($G_1(q), \ldots, G_8(q) \in \mathcal{RH}_\infty$) are generated as aforementioned,

$$Q = \begin{bmatrix} 0 & 0 & 0 & \beta G_7(q) \\ G_1(q) & 0 & \alpha G_6(q) & 0 \\ G_4(q) & G_2(q) & 0 & 0 \\ G_5(q) & 0 & G_3(q) & 0 \end{bmatrix}, P = \begin{bmatrix} G_8(q) \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.46}$$

and α, β are positive real numbers that need to be tuned to guarantee stability, $\tilde{G}_6 \triangleq \alpha G_6$, $\tilde{G}_7 \triangleq \beta G_7$. We start with the inner loop (labeled as "loop 1" in Figure 3.5) that is formed by the feedback item $\tilde{G}_6$, and applied the *small gain theorem* to the interconnected system shown in Figure 3.6a, which tells to choose $\alpha < 1/(\|G_2\|_\infty \|G_6\|_\infty)$. We then redraw Figure 3.5 to remove the feedback path of $\tilde{G}_6$, as shown in Figure 3.6b, and the resultant whole block diagram turns to be Figure 3.6c, where $T_1 = \frac{G_1 + G_4 \tilde{G}_6}{1 - G_2 \tilde{G}_6}$. Next we compute the lump-sum transfer function of all feedforward paths from y_1 to y_4 and present the following interconnected system in Figure 3.6d, where $T_2 = (T_1 G_2 + G_4) G_3 + G_5$. By applying the *small gain theorem*, we choose $\beta < 1/(\|T_2\|_\infty \|G_7\|_\infty)$. As demonstrated by this example, the idea is to tune the feedback component sequentially, from inner loops (e.g., "loop 1" in Figure 3.5) to outer loops (e.g., "loop 2"), using the *small gain theorem* to guarantee the internal stability.

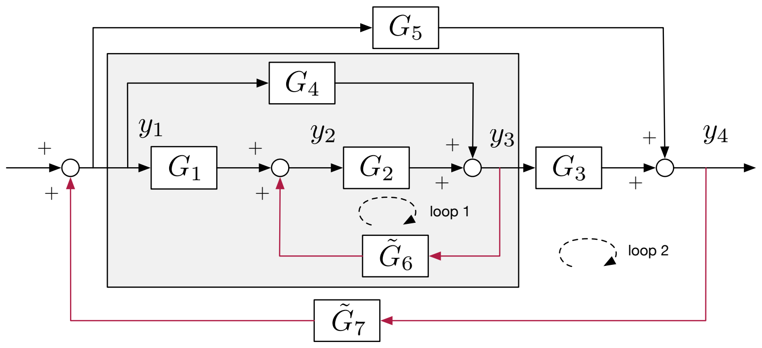
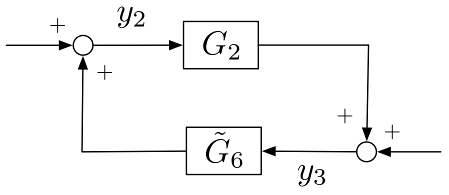
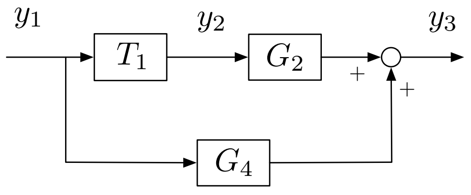


Figure 3.5. Block diagram of example (3.46) (neglecting inputs).

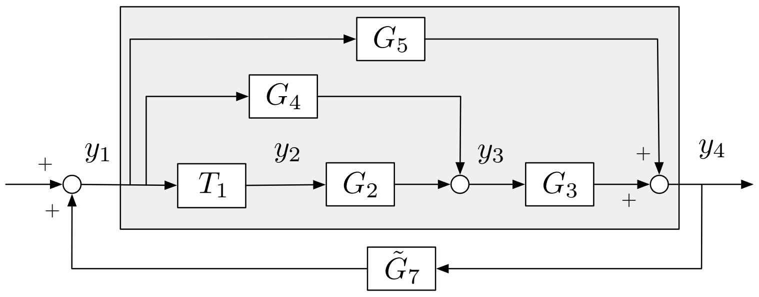
This example uses the block diagrams to explain the procedure to apply the *small gain theorem*. In practice, the signal-flow graph is a better choice to represent complicated interconnections and helps to apply *Mason's gain formula* to automate the computation of interconnected transfer functions. In our simulation, to ease the computation, up to networks of 20 nodes (see experiments for Figure 3.7b), we add at most 3 feedback paths (since the path/loop finding is an NP problem and cost considerable time). Moreover, to simply computation, these feedback loops are either non-touching (no common nodes) or one is "contained" by the other (i.e., all the nodes in loop 1 appear in loop 2). Due to page limits, the general algorithms will be present in another paper to handle arbitrary feedback and feasibility of network stabilization. Before moving to the inference part, there is one implementation detail in MATLAB deserving to be shared. When you use *control system*



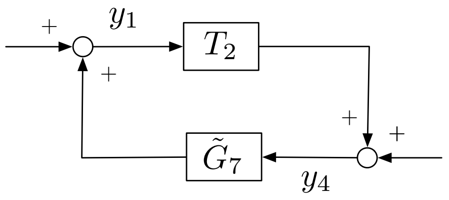
(a) lump-sum system



(b) intermediate subsystem



(c) intermediate system



(d) lump-sum system

Figure 3.6. List of intermediate systems that assist the description of the whole procedure of stabilizing ARX networks in model generation.

toolbox in MATLAB and deal with many (e.g., $p >= 10$) interconnections (connect in series or parallel, feedback), it may not be a good idea to compute the lump-sum transfer functions first, e.g., T_2 in Figure 3.6d, and then compute their infinity norms to apply the *small gain theorem*. The reason is that, even if at each step you guarantee the subsystems are correctly stabilized, the numeric errors in model interconnection using *control system toolbox* may lead to the resultant system being unstable (in theory it should be stable) and you are no longer able to continuing applying the *small gain theorem*. The solution used in our simulation is that, instead of computing the lump-sum transfer function, we compute the infinity norm of each transfer function (e.g., T_1, G_2, G_3, G_4, G_5 for T_2) first and then use *Mason's gain formula* to compute an upper bound of $\|T_2\|_\infty$.

In the benchmark studies on ARX networks, we test both iterative reweighted l_1 method (labeled as "GIRL1" in Table 3.1 and Figure 3.7) and sparse Bayesian learning (labeled as "GSBL") for group sparsity. There are two test scenarios: 1) models with $p = 10$ (i.e. #nodes = 10) and different SNRs (SNR $= 0, 10, 20, 40$ dB); 2) models with SNR = 10 dB and different number of nodes $p = 5, 10, 15, 20$. In each test (with one given p and SNR), we generate on 50 ARX network models with random network structures. All models are set to the same sparsity density 0.2, i.e. the total number of nonzero entries in Q is $0.2p^2$. The input signals are independently and identically distributed Gaussian noise with zero mean and unit variance. In the test of GIRL1, the regularization parameter λ is set to as follows: $\lambda = 0.1, 0.1, 0.01, 0.001$ for SNR $= 0, 10, 20, 40$dB, respectively; and $\lambda = 0.05, 0.1, 0.1, 0.1$ for $p = 5, 10, 15, 20$. The parameter λ is chosen roughly among values in logarithmic scales for one model by reviewing the sparsity density of the inference results (assuming we have an expectation on how sparse the network should be), and then is used for all 50 models. One certainly can apply cross-validation or bootstrap methods to choose λ.

Considering performance indices for benchmark, we use the indices *precision* (Prec) and *true positive rate* (TPR),

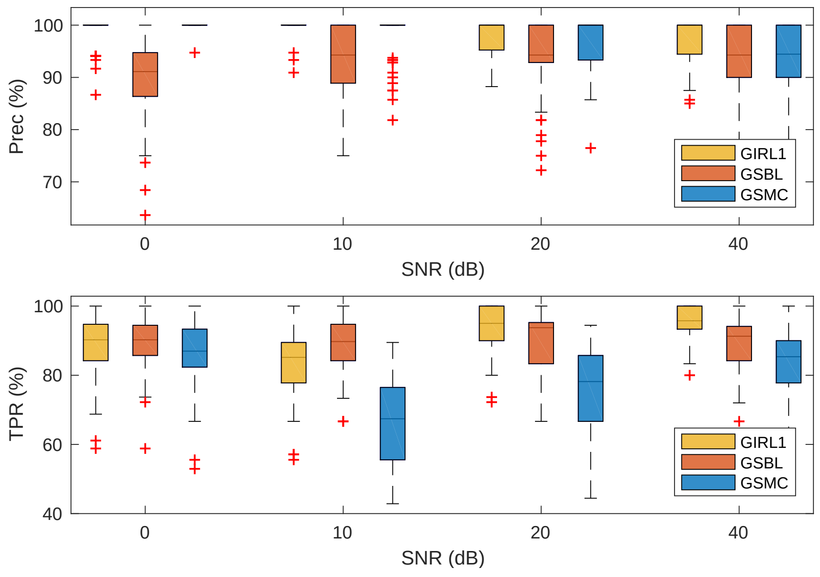
$$\text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

where TP (true positive), FP (false positive) and FN (false negative) are the standard concepts in the *Receiver Operating Characteristic* (ROC) curve or the *Precision Recall* curve (e.g. see Sahiner et al. (2017)). One may understand Prec as the percentage of correct arcs in the inferred network, and TPR as the percentage of correctly inferred arcs in the ground truth. As analogous to concepts *type-I error* and *type-II error* in statistics, The value of $(1-\text{Prec})$ is asserting the percentage of arcs in the inferred network that is absent (a false hit), and $(1-\text{TPR})$ is failing to assert the percentage of arcs in the ground truth that are present (a miss).

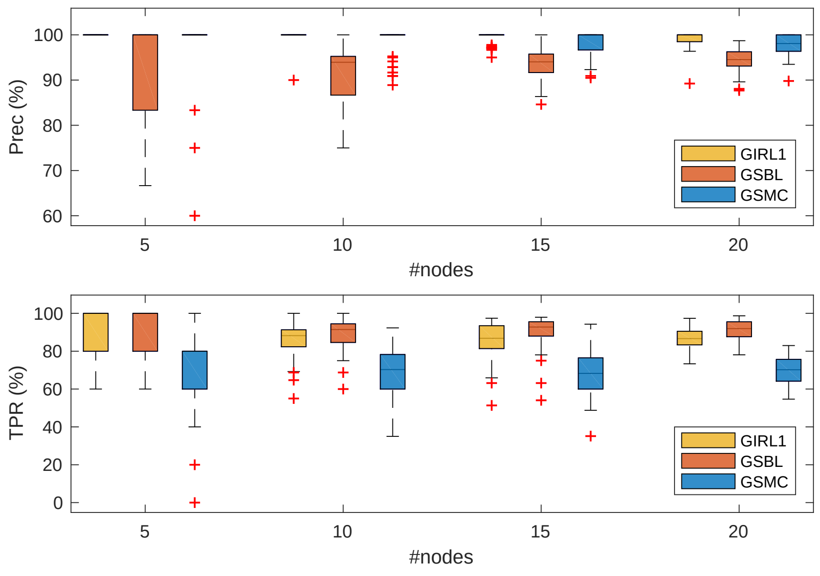
Table 3.1 summarizes the means and standard deviations (SD) of performance indexes of the inference results, whose box plots shown in Figure 3.7. Figure 3.7 tells that GIRL1 has better performance on Prec (larger means and smaller variances) that GSBL over different SNRs or #nodes; while the performance on TPR shows slighter weaker that GSBL. This is due to the choice of regularization parameter λ, which gives us freedom to balance between Prec and TPR. A larger λ mostly helps to improve Prec at the expense of TPR. The comparison of Prec or TPR of GIRL1 over different SNRs or #nodes may not be meaningful due to the usage of different λ's. This comparison for GSBL shows a slight increase of Prec or TPR when the SNR increases, which, however, is not as significant as our intuition might tell. The reason is that GSBL in theory can handle data with a reasonably large range of SNRs by estimating noise variances reliably. Nevertheless, one detail in practice deserves our attention. A threshold in GSBL needs to be tuned to prune γ's (labeled as "pGamma") in the EM iterations, which determines whether the parameters that specific λ corresponds to have been confidently zero in the probability sense. When dealing noisy data, a larger value of pGamma works better. In the Monte Carlo test of multiple #nodes, as shown in Figure 3.7b, one may observe large variances of performance indexes when #nodes is small. This is mostly due to the total number of arcs in the ground truth is so small that even inferring one arc wrongly leads to significant changes on Prec or TPR.

Table 3.1. Summary of inference results for ARX dynamic networks. Each statistic is computed from inference results of 50 random models. The label "GIRL1" refers to the iterative reweighted l_1 method for group sparsity, "GSBL" refers to Sparse Bayesian Learning and "GSM" refers to sampling methods. In each test, all methods use the same data set, except that "GSM" only uses 100 points. In the simulation of multiple SNRs, the number of nodes is set to 10 (i.e. $p = 10$); and in the case of multiple #nodes, the SNR is set to 10 dB. The lambdas for "GIRL1" are set to $\lambda = 0.1, 0.1, 0.01, 0.001$ for SNR $= 0, 10, 20, 40$dB, respectively; and $\lambda = 0.05, 0.1, 0.1, 0.1$ for $p = 5, 10, 15, 20$.

(mean±SD)		SNRs (dB)			
		0	10	20	40
Prec (%)	GIRL1	98.96 ± 2.77	99.58 ± 1.73	97.94 ± 3.25	97.34 ± 4.25
	GSBL	89.92 ± 8.16	93.05 ± 6.95	93.31 ± 7.25	93.35 ± 6.64
	GSMC	99.89 ± 0.74	97.57 ± 4.71	96.28 ± 5.39	94.20 ± 6.37
TPR (%)	GIRL1	88.05 ± 9.51	83.06 ± 10.40	92.95 ± 7.24	95.52 ± 5.32
	GSBL	89.29 ± 8.25	89.14 ± 8.45	89.92 ± 8.81	88.79 ± 8.36
	GSMC	84.64 ± 10.49	65.16 ± 12.48	75.85 ± 12.12	83.15 ± 10.47
(mean±SD)		#nodes			
		5	10	15	20
Prec (%)	GIRL1	100.00 ± 0.00	99.80 ± 1.41	99.45 ± 1.23	99.17 ± 1.74
	GSBL	94.47 ± 9.31	91.55 ± 6.87	93.66 ± 3.57	93.86 ± 4.45
	GSMC	98.33 ± 7.02	98.65 ± 3.03	98.21 ± 2.49	97.78 ± 2.23
TPR (%)	GIRL1	93.60 ± 10.25	86.23 ± 8.85	85.22 ± 9.85	86.82 ± 5.41
	GSBL	92.40 ± 11.35	89.05 ± 8.39	90.51 ± 8.59	90.89 ± 5.22
	GSMC	75.20 ± 20.82	69.67 ± 12.72	68.53 ± 12.38	69.50 ± 7.71



(a) multiple SNRs



(b) multiple #nodes

Figure 3.7. Performance of network inference using iterative reweighted l_1 method (labeled as "GIRL1"), sparse Bayesian learning (labeled as "GSBL") and the sampling method for group sparsity (labeled as "GSMC"). The data sets are generated from 50 models with different SNRs or numbers of nodes.

3.5.2 Dynamical structure functions

This section considers a Monte Carlo study of 50 runs of inference of *random stable sparse* networks (DSFs) with 40 nodes. In regard to the adjective words for the DSFs, here are further explanations:

- *random*: the DSF model in each run are randomly chosen (both network topology and model parameters);
- *stable*: the DSF model is stable, i.e. all poles of Q, P and H have negative real parts, and all transmission zeros of $(I-Q)$ have negative real parts; (see (Yue, 2018, chap. 2.3))
- *sparse*: the number of arcs of the network is much less than that of a complete digraph.

The numerical example emulates the applications in practice, where the underlying systems evolves continuously in time and the proposed method uses parametric approaches to estimate network structures. Hence, we simulate the continuous-time DSF models and then sample the simulated signals with a chosen sampling frequency to acquire measurements for later network reconstruction. More details on the procedure is presented as below:

1) The DSF model will be simulated via its state space realization (both in continuous time). Then we start with generation of random state space models. First we randomly generate highly sparse stable A-matrices (of dimension 80×80) for the state space representations, and $B = [0, \dots, 1, \dots, 0]^T$, $C = [I_{20\times 20}\ 0_{20\times 20}]$, $D = 0$. The systems in replica are obtained by perturbing nonzero entries in the A-matrix.

2) The ground truth of the networks are calculated by the definition of DSF using (A, B, C, D) (see Goncalves and Warnick (2008)).

3) A step signal is chosen to be the input[3], and each state variable is perturbed by a Gaussian i.i.d. (i.e. process noises). The replica data is acquired by randomly perturbing non-zero elements in A and performing simulation. The stochastic differential equation is numerically solved by using `sim.m` (choosing the Euler-Maruyama method) from *system identification toolbox* in MATLAB. The sampling frequency is chosen to be 40 times larger than the critical frequency of system aliasing (see Yue et al. (2016a)).

The setup of DSF models makes network inference particularly challenging. In these networks, there exist many feedback loops, whose sizes are quite random. Moreover, we fill nonzero values in the position $A_{i,i+1}$ of the A-matrix to ensure each network will not degenerate into a family of separate small ones.

[3]Here we choose to have only one input and use a step signal to simulate the biological data. In biological experiments, we usually do not have many controlled inputs, and most of them are simple signals, like fixed temperatures, adding/removing light, fixed pH values, etc.

As known in biological data analysis, time series are usually of low sampling frequencies and have limited numbers of samples. To address the importance of these factors, we run network inference methods (using ARX) over a range of values, shown in Figure 3.8. The sampling frequency is critical for applying discrete-time approaches for network inference, since the network topology from discrete-time systems will more and more different from the ground truth that is defined by the underlying continuous-time systems, with the decrease of sampling frequencies. The rule of thumb is choosing the sample frequency that is at least 10 time faster than the critical sampling frequency of system aliasing (e.g. $f_s/4$ in Figure 3.8 is the least suggested value). The sparsity is to handle the effect of noise. The simulation tells us that the value that is at least four times larger than the number of unknown parameters in estimation is a fair choice for the number of samples in network inference.

Another comment is for the “trade-off” between type-I and type-II errors when selecting regularization parameters λ. In theory, there could be an optimal value of λ that gives small values of both type-I and type-II errors. However, in practice, type-I error is more critical in the sense that it has to be small enough to keep results useful. Otherwise, even if type-II error is small, the result will predicate too many wrong arcs to be useful in applications. As a rule implied from Figure 3.8, in biological practice, we may choose an aggressive value of λ to make sure that we could have most predictions of arcs correctly; then, if more links need to be explored, we could decrease λ to get more connections covered.

3.6 Conclusions

This work discusses dynamic network reconstruction from heterogeneous datasets in the framework of dynamical structure function (or P. Van den Hof’s network representation) has been discussed. It has been addressed that linear dynamic network reconstruction can be formulated as identification of DSFs with sparse structures. To take advantage of heterogeneous datasets from multiple experiments, the proposed method integrates all datasets in one regression form and resorts to group sparsity to guarantee network topology to be consistent over replica. To solve the cardinality optimization problem, the treatments using classical convex approximation, SBL and sampling-based methods have been introduced and extended. The numerical examples have shown the performance of methods and revealed several issues that deserve our consideration in applications.

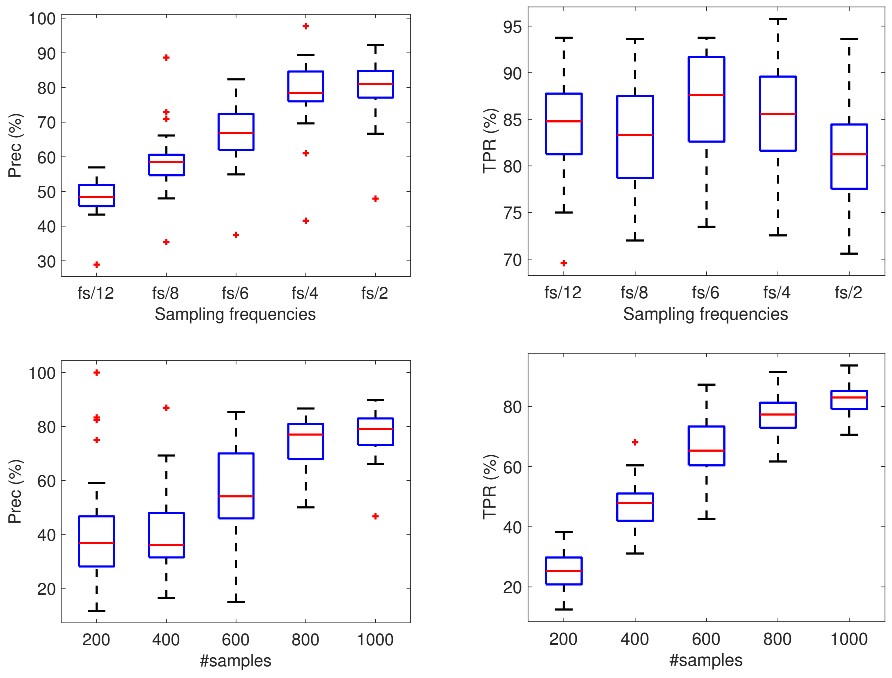


Figure 3.8. Performance of the proposed method on 50 random networks. The value of λ is chosen by performing cross-validation on one network. Choosing a larger λ could further decrease the Type-I error; however, the inferred networks would lose their basic profiles, tending to be simple topologies, like unconnected trees or lines. The sampling frequency f_s is the base value used in system simulations. The data used in inference are re-sampled from the simulated signals. Here we use 2 replica datasets, e.g. "#samples $=$ 800" implies each dataset has 400 samples. The iterative reweighted l_1 method is used to achieve group sparsity.

Chapter 4

Low Sampling Frequencies: System Aliasing

There have been studies focusing on identification of continuous-time systems, e.g., (Garnier et al., 2003; Ljung and Wills, 2010). However, most methods request a high sampling frequency to guarantee certain simplifications on theoretical deduction or numerical calculations. Moreover, choosing too low sampling frequencies may trigger the problem of "system aliasing", that is, multiple continuous-time systems produce the exactly same output samples, meanwhile they may have different network structures. What's more, it becomes particularly challenging in theory and numerical computation to enhance sparsity on the network structures of continuous-time models. However, this is inevitable considering the current techniques of biological time-series data acquisition.

This chapter first reveals the challenges due to the low sampling frequency by examples in Section 4.2 and then present a definition of *system aliasing*. A Nyquist-Shannon-like sampling theorem was presented in Section 4.3 to determine the minimal sampling frequency that avoids the effect of system aliasing. Section 4.4 presents an algorithm to reconstruct sparse networks using low-sampling-frequency data provided with no system aliasing. The case of system aliasing is discussed in Section 4.5.

4.1 Problem Description

Consider a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,\infty)}, \mathbb{P})$, where the filtration is always assumed to be complete. Let $\{w(t) : t \geq 0\}$ be the n-dimensional standard $\mathcal{F}_t$-Brownian motion. The physical plant/process in our study, as a dynamical system in continuous time, is modeled by the following stochastic differential equation

$$\mathrm{d}x = Ax\,\mathrm{d}t + Bu\,\mathrm{d}t + R\mathrm{d}w, \tag{4.1}$$

where $A \in \mathbb{R}^{n\times n}$ is stable, $R \in \mathbb{R}^{n\times n}$ is symmetric and semi-positive definitive, the initial $x(t_0)$ is a Gaussian random variable with mean m_0 and variance R_0, $t_0 \geq 0$, and $w(t)$ is interpreted as *disturbance* on the state variables (or called *process noise*). The solution to (4.1) is an $\mathcal{F}_t$-adapted n-dimensional stochastic process $x(t) = (x_1(t), \cdots, x_n(t))_{t\geq t_0}$ such that

$$x(t) = x(t_0) + \int_{t_0}^{t} (Ax(s) + Bu(s))\, \mathrm{d}s + \int_{t_0}^{t} R\mathrm{d}w(s),$$

where $x(t_0) \in \mathcal{F}_{t_0}$ and see Gall (2016) for the definition of stochastic integral. It is assumed that $x(t_0), w(t)$ are independent. The solution $x(t)$ is *strong*, that is, $x(t)$ is adapted to $\mathcal{F}_t^w := \sigma(w_u : u \leq t) \vee \sigma(\mathcal{N})$ (i.e. the complete σ-field generated by $\{w_u : u \leq t\}$; see Gall (2016) for details). An input signal $\{u(t), t \geq t_0\}$ has been applied to the system, and the output y of the system is observed at the discrete times $t_0, t_1, ..., t_N$,

$$y(t_k) = Cx(t_k) \tag{4.2}$$

where $C = [I\ 0] \in \mathbb{R}^{p\times p}$, $p \leq n$, $t_k \triangleq t_0 + kh$ and $h > 0$ is the sampling period. Here the measurement noise is not included mainly due to that we do not know how to give a definition of DSF from state-space representations with measurement noises. The stochastic difference equation that relates the values of the state variable x in (4.1) at the sampling instants (Åström, 2012, p. 82-85); (Garnier and Wang, 2008, chap. 2), is given by

$$x(t_{k+1}) = A_d x(t_k) + B_d u(t_k) + v(t_k), \tag{4.3}$$

where

$$A_d = \exp(hA), \quad B_d = \int_0^h \exp(sA)B\, \mathrm{d}s, \tag{4.4}$$

$\exp(\cdot)$ is the matrix exponential, and the Gaussian i.i.d. $v(t)$ has mean zero and covariance matrix

$$R_d = \int_0^h \exp(sA)R\exp(sA^T)\, \mathrm{d}s. \tag{4.5}$$

The *linear dynamic network* model[1] of (4.1) is given as

$$y(t) = Q(s)y(t) + P(s)u(t) + sH(s)w(t), \tag{4.6}$$

where $Q(s), P(s)$ and $H(s)$ are $p \times p$, $p \times m$ and $p \times p$ matrices of strictly-proper real-rational transfer functions respectively, defined from $(A, B, C, 0)$ in Goncalves and Warnick (2008); and s is the differential operator, i.e. $sx(t) = \mathrm{d}x/\mathrm{d}t$. The model (4.6) is called *dynamical structure function* (DSF), firstly proposed in Goncalves and Warnick (2008) and other variants in Van

[1] Recall Brownian motion is locally α-Hölder continuous ($\alpha < 1/2$) almost surely and nondifferentiable. Since Q, P and sH are proper, three terms in (4.6) are well defined and their meanings are given by stochastic integration.

den Hof et al. (2013); Weerts et al. (2018b). The *dynamic networks* $\mathcal{N} = (\mathcal{G}, f)$ can be defined from the DSF (4.6) by Definition 2.1, which graphically represents the interconnections between the output variables.

This chapter focuses on the full-state measurement case, i.e. $C = I$, where A in (4.1) tells the interconnections between elements of y since $Q(s) = (s - \operatorname{diag}(A))^{-1}(A - \operatorname{diag}(A))$. Indeed our study considers a "simple" case. However, due to the focus on low sampling frequencies, one will see it has been fairly complicated in later sections. Let measurements be $Y^N \triangleq [y(t_0),\ y(t_1),\ \ldots,\ y(t_N)]$, and $U^N \triangleq [u(t_0),\ u(t_1),\ \ldots,\ u(t_N)]$ (if having external inputs). We then describe the problem of network reconstruction in our study as follows:

Main Problem. Given signals Y^N (and U^N, if having inputs) in full-state measurement (i.e. $C = I$), with probably a large sampling period h and small N, develop methods to infer the dynamic network $\mathcal{N}$ (or the Boolean $\mathcal{G}$), assuming that the ground truth A is sparse.

Remark 4.1. There are two main questions we will answer in later sections:

- What is the minimal sampling frequency that we can tackle in theory? And is there a criterion to test sampling frequencies of the given signals?
- Due to low sampling frequencies, we have to resort to continuous-time models (i.e. A, B in (4.1)) to determine networks. How could we impose sparsity of network structures in the inference procedure?

Throughout the text, by default, we always deal with *primary* matrix functions, including exp (matrix exponential), log (matrix logarithm) and Log (the principal matrix logarithm in Theorem 4.3). *Primary* matrix functions refer to the ones defined via *Jordan Canonical Form* or equivalently via *Polynomial Interpolation*, *Cauchy Integral Theorem* (Higham, 2008a, chap. 1). The *primary* notion of matrix functions is of particular interest and the most useful in applications, e.g. (Higham, 2008a; Horn and Piepmeyer, 2003).

4.2 System Aliasing in Identification

4.2.1 Observations on matrix logarithm

Supposing that A_d has been perfectly estimated from samples, the estimate of the A matrix for the continuous-time system is straightforwardly calculated by solving

$$\exp(hA) = A_d. \tag{4.7}$$

via matrix logarithm. However, referring to Theorem A.1 (Higham, 2008a), the equation (4.7) has several (in fact infititely many) solutions. Let us review the following observations on (4.7) to see the troubles from low sampling frequencies (i.e. $1/h$).

Observation 1: With the increase of h, the Boolean structures of A (i.e. $\mathcal{G}$ determined from A) and $A_d - I$ become more and more different, as illustrated by Figure 4.1. The sampling frequency ($1/h$) deserves to be emphasized as a core factor in the categorization of different cases in our study:

- Case I: when h is "very small" such that A_d shares the same Boolean structure as $hA + I$. Indeed, one can see it by $\exp(hA) = I + hA + \frac{h^2}{2!}A^2 + \cdots$. Hence we can determine $\mathcal{G}$ by identifying discrete-time models;
- Case II: when h is "large" but the ground truth A is still the principle matrix logarithm of A_d;
- Case III: when h is "even larger" such that the ground truth A is no longer the principle logarithm of A_d.

The general network model (4.6) of Case I has been solved by discrete-time approaches , e.g. see (Chiuso and Pillonetto, 2012; Yue et al., 2018). Case II is what we mainly studied in this paper. We call both Case I and II *no system aliasing*, as defined and studied in later sections.

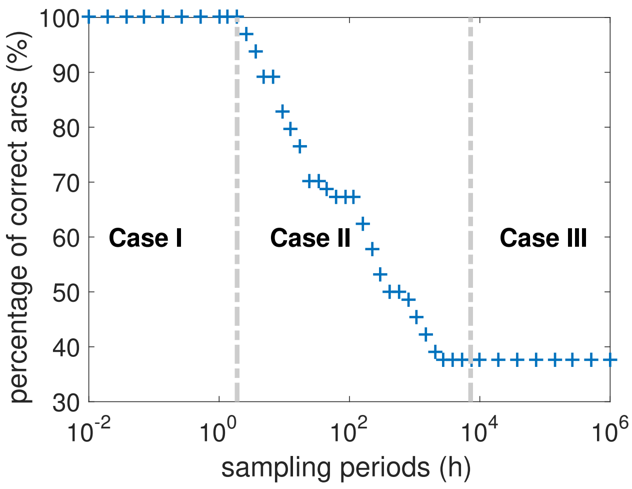


Figure 4.1. This example randomly chooses a sparse A and $A_d = \exp(hA)$. The Boolean networks are determined from A and $A_d - I$ by Definition 2.1 (treating A and $A_d - I$ as Q's), denoted by $\mathcal{G}(A), \mathcal{G}(A_d - I)$. We use $\mathcal{G}(A)$ as the ground truth. Then $\mathcal{G}(A_d - I)$ is compared with $\mathcal{G}(A)$ and the same arcs are labeled as the correct ones.

Observation 2: Provided with a sparse A, the corresponding A_d could be dense, as the example in Figure 4.2.

$$A = \begin{bmatrix} 0 & 0.0149 & 0.0790 & \boxed{0} \\ \boxed{0} & 0.1932 & 0.1741 & 0.0313 \\ 0 & 0.0325 & \boxed{0} & \boxed{0} \\ 0.0231 & 0.0020 & \boxed{0} & 0 \end{bmatrix} \qquad A_d = \begin{bmatrix} 1.0000 & 0.0178 & 0.0804 & \boxed{0.0003} \\ \boxed{0.0004} & 1.2164 & 0.1923 & 0.0346 \\ 0.0000 & 0.0359 & \boxed{1.0030} & \boxed{0.0005} \\ 0.0231 & 0.0024 & \boxed{0.0011} & 1.0000 \end{bmatrix}$$

Figure 4.2. $h = 1,\ A_d = \exp(hA),\ A = \operatorname{Log} A_d/h$.

Observation 3: Provided with a sparse A, the corresponding A_d can be also sparse. However, they have different Boolean structures (i.e. zeros at different positions), as shown in Figure 4.3.

$$A = \begin{bmatrix} \boxed{0} & 0 & 0 & 0.5303 \\ 0.8611 & 0 & 0.4849 & \boxed{0} \\ 0 & 0 & 0.3935 & 0 \\ 0.6714 & 0 & 0 & \boxed{0} \end{bmatrix} \qquad A_d = \begin{bmatrix} \boxed{1.1834} & 0 & 0 & 0.5624 \\ 0.9132 & 1.0000 & 0.5941 & \boxed{0.2352} \\ 0 & 0 & 1.4821 & 0 \\ 0.7120 & 0 & 0 & \boxed{1.1834} \end{bmatrix}$$

Figure 4.3. $h = 1,\ A_d = \exp(hA),\ A = \operatorname{Log} A_d/h$.

Observation 4: Even though the norm difference of A_{d1}, A_{d2} has been very small, their matrix logarithms, e.g. A_1 and A_2 in Figure 4.4, have significantly different Boolean structures.

$$\frac{\|A_{d2} - A_{d1}\|_2}{\|A_{d1}\|_2} = 0.0138\%$$

$$A_1 = \begin{bmatrix} \boxed{0} & \boxed{0} & 0.1190 & \boxed{0} \\ 0.4984 & \boxed{0} & 0.9597 & 0.3404 \\ 0 & 0 & \boxed{0} & \boxed{0} \\ \boxed{0} & 0.5853 & 0.2238 & 0 \end{bmatrix} \qquad A_2 = \begin{bmatrix} \boxed{0.0001} & \boxed{0.0001} & 0.1188 & \boxed{0.0001} \\ 0.4982 & \boxed{0.0002} & 0.9595 & 0.3403 \\ 0.0000 & -0.0000 & \boxed{0.0002} & \boxed{0.0001} \\ \boxed{0.0001} & 0.5851 & 0.2239 & -0.0000 \end{bmatrix}$$

Figure 4.4. $h = 1,\ A_1 = \operatorname{Log} A_{d1}/h$ and $A_2 = \operatorname{Log} A_{d2}/h$.

Observation 5: There exists more than one solution, e.g. A_1 and A_2 in Figure 4.5, and they have different Boolean structures.

Considering the problem formulation in Section 4.1, one may have already noticed the troubles on network reconstruction, which originate from the matrix logarithms, due to the low sampling frequency. The examples in Observation 1 clearly show that why we have to resort to the continuous-time system identification to infer network structures. Observation 2 and 3 tell that A and A_d do not share the same sparsity when h is large. Observation 4 points out that the Boolean structures of the principle logarithms of two A_{d1} and A_{d2} close in matrix norms could be significantly different. The example on Observation 5 shows an even worse case: the sample period is so large that the principle matrix logarithm is no longer A, which appears as other branches of matrix logarithm of A_d, in which no robust algorithm has yet been available.

Remark 4.2. In a sum of the above observations, it tells us that, in the inference of $\mathcal{G}$ or $\mathcal{N}$,

$$A_1 = \begin{bmatrix} -3.7147 & -0.1778 & 0.1625 & 0.2690 \\ \boxed{-0.8983} & -5.4242 & -0.5625 & 0.9199 \\ 1.7815 & -0.2501 & \boxed{-4.7786} & -1.1221 \\ 0.8726 & 0.0043 & -0.6277 & -3.9162 \end{bmatrix}$$

$$A_d = \begin{bmatrix} 0.0300 & -0.0025 & 0.0023 & 0.0038 \\ -0.0127 & 0.0059 & -0.0079 & 0.0130 \\ 0.0252 & -0.0035 & 0.0150 & -0.0158 \\ 0.0123 & 0.0001 & -0.0089 & 0.0272 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -9.1872 & -1.7824 & -2.9906 & 3.1019 \\ \boxed{0} & -5.3854 & -8.4002 & -4.0112 \\ 16.4473 & 13.2820 & \boxed{0} & -7.4828 \\ -13.7279 & 8.6466 & -2.7852 & -3.2612 \end{bmatrix}$$

$$\begin{tikzcd} A_1 \arrow[dr, dash, bend left] & \\ & A_d \\ A_2 \arrow[ur, dash, bend right] & \end{tikzcd}$$

Figure 4.5. $h = 1$, $\exp(hA_1) = \exp(hA_2)$, $A_1 = \operatorname{Log} A_d/h$, and A_2 is the ground truth A.

- $\mathcal{G}$ should be determined from A instead of A_d, when h is large (w.r.t. A); (see Observation 1, 4)
- the sparsity penalty has to be imposed on A directly instead of A_d, when h is large (w.r.t. A); (see Observation 2, 3, 4)
- the A matrix should be estimated directly instead of via taking matrix logarithm of A_d, in the presence of noise and with a limited length of signals. (see Observation 4)

And we summarize the tasks in our study as follows:

- finding out the critical frequency (i.e. the 2nd vertical dash line in Figure 4.1) in theory that demarcates Case III (see Observation 1, 5), and
- developing inference methods for Case II (also being valid for Case I, which, however, can be more efficiently solved by discrete-time approaches) with the concerns aforementioned.

4.2.2 Definition of system aliasing

As shown in Section 4.2.1, the A matrix has to be identified in network reconstruction when the sampling frequency is low. In this scenario, a "good" case is that the ground truth A is the principle matrix logarithm of A_d (i.e. Case II); otherwise, it becomes particularly challenging (i.e. Case III), e.g. Figure 4.5. To clarify this classification, we consequently present an important concept in network reconstruction with low sampling frequencies, "*system aliasing*".

Let $\operatorname{vec}(X)$ denote the vectorization of the matrix X formed by stacking the columns of X into a single column vector; and $\operatorname{ivec}(\cdot)$ is defined by $\operatorname{ivec}(\operatorname{vec}(X)) = X$. $\operatorname{Im}(x)$ denotes the imaginary part of the complex number or vector x.

Definition 4.1.

$$\begin{aligned}\mathscr{E}(A, M, h, \mathscr{S}) = \Big\{ & A^* \in \mathbb{R}^{n\times n} : \ M \in \mathbb{R}^{n^2\times n^2}, h \in \mathbb{R}, \\ & A^* = \arg\min_{\tilde{A}\in\mathscr{S}} \|M\,\mathrm{vec}(\exp(hA)) - M\,\mathrm{vec}(\exp(h\tilde{A}))\|_2 \Big\},\end{aligned}$$

where $\mathscr{S} \subseteq \mathbb{R}^{n\times n}$ contains A.

With this general notation, we present a definition of *system aliasing* only in terms of the A matrix in state-space representations and the sampling period h, which does not depend on specific identification methods or data.

Definition 4.2 (System aliasing). Given $A \in \mathscr{S}$ and $h \in \mathbb{R}_+$, if there exists $\hat{A} \neq A \in \mathscr{E}(A, I, h, \mathscr{S})$ and $\hat{A}$ is called *system alias* of A with respect to $\mathscr{S}$. By default, choose $\mathscr{S} = \mathscr{S}_A := \{\tilde{A} \in \mathbb{R}^{n\times n} : \max\{\mathrm{Im}(\mathrm{eig}(\tilde{A}))\} \leq \max\{\mathrm{Im}(\mathrm{eig}(A))\}\}$.

We are particularly interested in $\mathscr{E}(A, I, h, \mathscr{S}) = \{A\}$, i.e. there is no problem of *system aliasing*. Note that the concept of *system aliasing* does not depend on specific data. It only depends on system dynamics (the A matrix in (4.1)) and sampling frequencies. If the M matrix is specifically constructed by data instead of I, $\mathscr{E}(A, M, h, \mathscr{S}) = \{A\}$, where A denotes the ground truth, tells that the underlying system is identifiable from the given data (see (Yue et al., 2016a, Sec. III-B)). Obviously if we have system aliasing for the system with a specific sampling frequency, without extra prior information on A, the system is always not identifiable.

4.3 No System Aliasing: the Minimal Sampling Frequency

Provided with the definition of *system aliasing*, a question comes first: what (A, h) leads to no system aliasing, i.e. $\mathscr{E}(A, I, h, \mathscr{S}_A) = \{A\}$.

4.3.1 The minimal sampling frequency

To answer this question, we need to introduce a theorem on matrix logarithm.

Theorem 4.3 (principal logarithm (Higham, 2008a, Thm. 1.31)). *Let $P \in \mathbb{C}^{n\times n}$ have no eigenvalues on $\mathbb{R}^-$. There is a unique logarithm A of P all of whose eigenvalues lie in the strip $\{z : -\pi < \mathrm{Im}(z) < \pi\}$. We refer to A as the principal logarithm of P of write $A = \mathrm{Log}(P)$. If P is real then its principal logarithm is real.*

To make the principal matrix logarithm $\mathrm{Log}(\cdot)$ be well-defined, we always assume that $\exp(hA)$ has no negative real eigenvalues. Let $\mathscr{G}(h) = \{z \in \mathbb{C} : -\pi/h < \mathrm{Im}(z) < \pi/h\}$. By Theorem 4.3 and A.1, it always holds that $\mathrm{Log}(\exp(hA))/h \in \mathscr{E}(A, I, h, \mathscr{S}_A)$. To avoid *system aliasing*, it implies that $\mathrm{Log}(\exp(hA))/h = A$, i.e. $\mathrm{eig}(A) \in \mathscr{G}(h)$. It is summarized as the following lemma.

Lemma 4.4. *Let $A_d = \exp(hA), h \in \mathbb{R}_+$, which has no negative real eigenvalues, and $\hat{A} = \mathrm{Log}(A_d)/h$. Then $A = \hat{A}$ (i.e. $\mathscr{E}(A, I, h, \mathscr{S}_A) = \{A\}$) if and only if* $\mathrm{eig}(A) \in \mathscr{G}(h)$.

Given no other information on the system, consider the identification problem of A using full-state measurement. It is necessary to decrease the sampling period h until the ground truth falls into the strip of $\mathscr{G}(h)$, and then the principal logarithm refers to the ground truth A, as illustrated in Figure 4.6. Otherwise, we would be bothered by *system aliases* of A and be unable to make a decision, unless we know extra prior information on A.

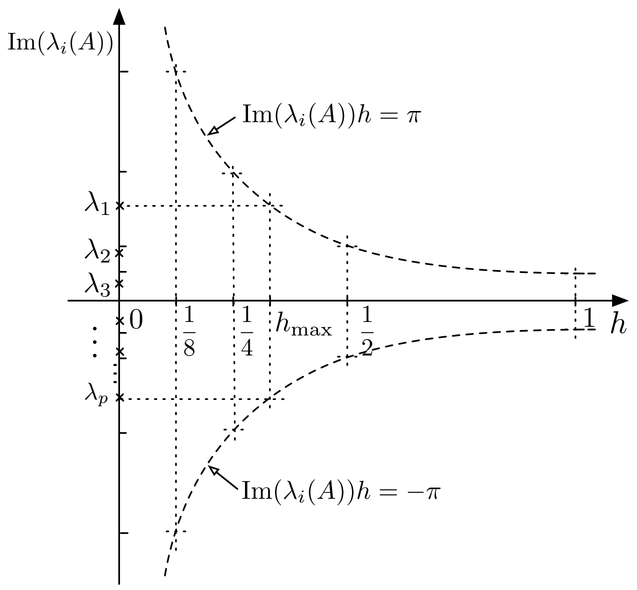


Figure 4.6. The imaginary parts of all eigenvalues of A must lie into $(-\pi/h, \pi/h)$. $\lambda_i(\cdot)$ denotes the i-th eigenvalue of A in Theorem A.1. The symbols "$\times$" denote the locations of $\mathrm{Im}(\lambda_i(A))$. $h_{\max}$ is the maximal sampling period that allows taking principal logarithms to estimate A, without facing troubles from system aliasing.

Theorem 4.5 (Nyquist-Shannon-like sampling theorem). *Consider the equidistant sampling. To uniquely reconstruct A of a continuous-time system from A_d of its corresponding discrete-time system by taking the principal matrix logarithm, the sampling frequency ω* (rad/s) *must satisfy*

$$\omega \geq 2 \max \{ | \, \mathrm{Im} \, (\lambda_i(A)) \, |, \ i = 1, \ldots, n \} \, .$$

Equivalently, the sampling period h (i.e. $2\pi/\omega$) should satisfy

$$h \le \min\{\pi/|\operatorname{Im}(\lambda_i(A))|,\ i = 1, \dots, n\}.$$

Proof. The result immediately follows by verifying the condition $\operatorname{eig}(A) \in \mathcal{G}(h)$ in Lemma 4.4. □

The meaning of Theorem 4.5 in continuous-time system identification can be understood by analogy with the *Nyquist-Shannon sampling theorem* in signal processing. The *Nyquist-Shannon sampling theorem* gives conditions on sampling frequencies, by looking at spectral information of signals, under which the continuous signal can be uniquely reconstructed from its discrete-time signal. As an analogy, Theorem 4.5 addresses that the continuous-time LTI system can be uniquely reconstructed from the discrete-time system under a condition that is built based on the spectrum of the A matrix.

4.3.2 Test criteria for the cases without inputs

Now we would like to show a property of matrix exponential and logarithm, which further leads to a test criterion on system aliasing. The test criterion is summarized in Section 4.3.3.

Lemma 4.6. *Considering $h_1, h_2 \in \mathbb{R}_+$ and $A \in \mathbb{R}^{n\times n}$, let $\hat{A}$ be defined by $\hat{A} = \operatorname{Log}(\exp(h_1 A))/h_1$. Then $\exp(h_2 A) = \exp(h_2\hat{A})$ if and only if $h_2/h_1 \in \mathbb{N}$ or $A = \hat{A}$.*

Proof. Let $A_d := \exp(h_1 A)$, which has the Jordan canonical form (A.1) (i.e. let P in Theorem A.1) be A_d). By Theorem A.1 and A.2, we have

$$\begin{aligned} h_1 A &= Z\operatorname{diag}(L_1^{j_1}, \dots, L_p^{j_p})Z^{-1}, \\ h_1\hat{A} &= Z\operatorname{diag}(L_1^0,\ \dots, L_p^0\ )Z^{-1}. \end{aligned}$$

To compare $\exp(h_2\hat{A})$ with $\exp(h_2 A)$, we need to find their Jordan canonical form by the definition of matrix exponential. To calculate the eigenvalues of $h_2\hat{A}$, consider the determinant $|\hat{\mu}I - h_2\hat{A}| = 0 \Leftrightarrow |\hat{\mu}'I - h_1\hat{A}| = 0$, where $\hat{\mu}' \triangleq h_1/h_2\hat{\mu}$ and $|\hat{\mu}'I - h_1\hat{A}| = |\hat{\mu}'I - \operatorname{diag}(L_1^0, \dots, L_p^0)|$. It is equivalent to solve p equations $|\hat{\mu}'_k I_k - L_k^0| = 0$ $(k = 1, \dots, p)$, where I_k denotes the identity matrix of the dimension compatible with L_k^0. It yields that

$$\begin{aligned} |\hat{\mu}'_k I_k - L_k^0| &= |\hat{\mu}'_k I_k - \log(J_k(\lambda_k))| \\ &= \left|\hat{\mu}'_k I_k - \begin{bmatrix} \log(\lambda_k) & \log'(\lambda_k) & \cdots & * \\ & \log(\lambda_k) & \ddots & \vdots \\ & & \ddots & \log'(\lambda_k) \\ & & & \log(\lambda_k) \end{bmatrix}\right| \\ &= (\hat{\mu}'_k - \log(\lambda_k))^{m_k} = 0, \end{aligned}$$

where $J_k, \lambda_k, j_k, m_k, I_{m_k}$ are given in (A.1), (A.3); and hence $\mu_k = h_2/h_1 \log(\lambda_k)$ with geometric multiplicity m_k. Similarly for $h_2 A$, consider $|\mu'_k I_k - L_k^{j_k}| = |\mu'_k I_k - \log(J_k(\lambda_k)) - 2 j_k \pi i I_{m_k}| = 0$, where $\mu'_k \triangleq h_1/h_2 \mu_k$, μ is the eigenvalues of $h_2 A$, the integer j_k is given in (A.3), and i is the imaginary unit. It yields $m_k = h_2/h_1(\log(\lambda_k) + 2 j_k \pi i)$ with multiplicity m_k. Considering the special forms of $L_k^{j_k}$, we have the following Jordan decomposition

$$\begin{aligned}\operatorname{diag}(L_1^0, \dots, L_p^0) &= U \operatorname{diag}(J_1(\hat{\mu}_1), \dots, J_p(\hat{\mu}_p)) U^{-1},\\ \operatorname{diag}(L_1^{j_1}, \dots, L_p^{j_p}) &= U \operatorname{diag}(J_1(\mu_1), \dots, J_p(\mu_p)) U^{-1},\end{aligned}$$

where $J_k(\hat{\mu}_k)$ and $J_k(\mu_k)$ denote the corresponding Jordan blocks. Therefore, $\exp(h_2 A) = \exp(h_2 \hat{A})$ is equivalent to $\exp(J_k(\mu_k)) = \exp(J_k(\hat{\mu}_k))$ for any $k = 1, \dots, p$, which implies $\exp(h_2/h_1 2 j_k \pi i) = 0$. It leads to the conditions: $h_2/h_1 \in \mathbb{N}_+$, or $j_k \equiv 0, \forall k$ (i.e. $\hat{A} = A$). □

Proposition 4.7. *Consider the dynamical system (4.1) without inputs (i.e. $B = 0$), and two sampling periods $h_1, h_2 \in \mathbb{R}_+$ such that $h_2/h_1 \notin \mathbb{N}$. Let $\hat{A} = \operatorname{Log}(\exp(h_1 A))/h_1$ and $\hat{A} \neq A$. The one-step prediction errors w.r.t. h_2 are defined as*

$$\begin{aligned}\epsilon(t_k) &= x(t_k) - \exp(h_2 A) x(t_{k-1}),\\ \hat{\epsilon}(t_k) &= x(t_k) - \exp(h_2 \hat{A}) x(t_{k-1}).\end{aligned}$$

Assuming that $\mathbb{E}(x(t_{k-1})) \neq 0$, it yields

$$\mathbb{E}(\epsilon(t_k)) = 0, \quad \mathbb{E}(\hat{\epsilon}(t_k)) \neq 0.$$

Proof. Considering the dynamical system (4.1), it is obvious that $\mathbb{E}(\epsilon(t_k)|x(t_{k-1})) = 0$. Now we evaluate the other expectation

$$\begin{aligned}\mathbb{E}(\hat{\epsilon}(t_k)) &= \mathbb{E}\left(x(t_{k+1}) - \exp(h_2 \hat{A}) x(t_k)\right)\\ &= \mathbb{E}\left(\left(x(t_{k+1}) - \exp(h_2 A) x(t_k)\right) + \left(\exp(h_2 A) - \exp(h_2 \hat{A})\right) x(t_k)\right)\\ &= 0 + \left(\exp(h_2 A) - \exp(h_2 \hat{A})\right) \mathbb{E}(x(t_k)) \neq 0,\end{aligned}$$

by Lemma 4.6. □

4.3.3 Test criteria for the cases with inputs

We have similar results for the case with inputs, as stated in Proposition 4.8, where we no longer require $\mathbb{E}(x(t_k)) \neq 0$ due to the benefits from inputs. Meanwhile, according to the condition (4.8), it is possible that a carefully designed input signal invalidates the test criterion that is built by evaluating $\mathbb{E}(\hat{\epsilon}(t_k))$, which in practice may not be a problem.

Proposition 4.8. *Consider the dynamical system (4.1), and two sampling periods $h_1, h_2 \in \mathbb{R}_+$ such that $h_2/h_1 \notin \mathbb{N}$. Let $f(A, B, h, t_k) \triangleq x(t_k) - \exp(hA) x(t_{k-1}) - \int_0^h \exp(sA) B \, \mathrm{d}s \, u(t_{k-1})$,*

$\hat{A} = \mathrm{Log}(\exp(h_1 A))/h_1$ *and* $\hat{B}$ *be a value such that* $\mathbb{E}(f(\hat{A}, \hat{B}, h_1, t_k)|x(t_{k-1}), u(t_{k-1})) = 0$ *for all* t_k*. The one-step prediction errors w.r.t.* h_2 *are defined as* $\epsilon(t_k) = f(A, B, h_2, t_k)$*,* $\hat{\epsilon}(t_k) = f(\hat{A}, \hat{B}, h_1, t_k)$*. If*

$$\mathbb{E}\Big(\big(\exp(h_2 A) - \exp(h_2 \hat{A})\big)x(t_k) \ + \int_0^{h_2} \big(\exp(sA)B - \exp(s\hat{A})\hat{B}\big)\mathrm{d}s\, u(t_k)\Big) \neq 0, \qquad (4.8)$$

we have

$$\mathbb{E}(\epsilon(t_k)) = 0, \quad \mathbb{E}(\hat{\epsilon}(t_k)) \neq 0.$$

Proof. The proof follows trivially by evaluating the expectation $\mathbb{E}(\hat{\epsilon}(t_k))$. We no longer require $\mathbb{E}(x(t_k)) \neq 0$ since we can take advantages of $u(t_k)$ to satisfy (4.8). In the cases with $\mathbb{E}(x(t_k)) = 0$ and non-zero inputs in expectation $\mathbb{E}(u(t_k)) \neq 0$, assuming $A, \hat{A}$ are non-singular, the condition (4.8) can be further simplified as

$$\big(\exp(h_2\hat{A}) - I\big)\big(\exp(h_1\hat{A}) - I\big)^{-1}\big(\exp(h_1 A) - I\big) \neq \exp(h_2 A) - I. \qquad (4.9)$$

The simplification follows from (4.8) by noticing $\int_0^h e^{xs}\mathrm{d}s = (e^{hx} - 1)/x$ and A commutes with its matrix functions. □

The basic idea behind Proposition 4.7 and 4.8 can be understood by Figure 4.8, where the output prediction of $\hat{A}$ (that is estimated from samples in h_1) presents different values from that of A in another sampling period h_2 and it results in that the expectation of one-step prediction errors is no longer zero. The test criterion on system aliasing is illustrated as Figure 4.7 and summarized as follows:

Test criterion. Identify A, B by PEM or ML (denoting the estimates by $\hat{A}, \hat{B}$) assuming no system aliasing under the sampling period h_1, i.e. $\hat{A}$ asymptotically converges to $\mathrm{Log}(\exp(h_1 A))$. Choose another sample period h_2 such that $h_2/h_1 \notin \mathbb{N}_+$, and sample by h_2 the system responses with non-zero initial conditions or non-zero inputs (assuming (4.8) is satisfied). Use $\hat{A}, \hat{B}$ to calculate the one-step prediction errors $\{\epsilon(t_k)\}$. Perform *t-test* to obtain the *p-value* to make decisions, where the null hypothesis is that $\{\epsilon(t_k)\}$ comes from a normal distribution with mean zero and unknown variance. Rejecting the null hypothesis implies the existence of system aliasing.

4.4 No System Aliasing: Sparse Network Reconstruction

Considering the systems given by (4.1) and (4.2), the likelihood function is determined by the multiplication rule for conditional probability

$$p(Y^N|\vartheta) \triangleq p(y(t_N), \ldots, y(t_1)|\vartheta) = p(y(t_N)|y(t_{N-1}), \vartheta)\, p(y(t_{N-1})|y(t_{N-2}), \vartheta) \cdots p(y(t_1)|\vartheta),$$

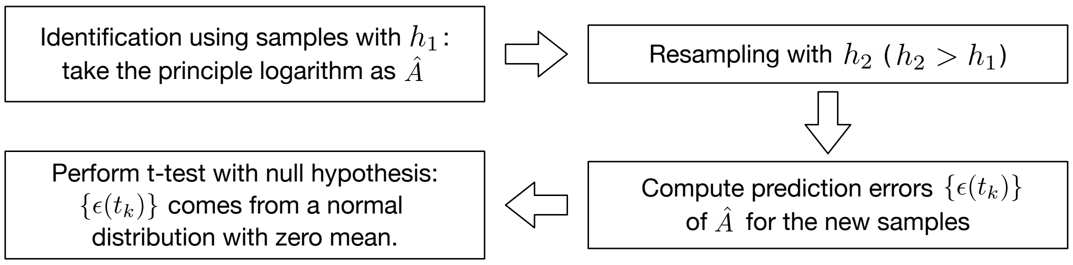

Identification using samples with h_1: take the principle logarithm as $\hat{A}$
Resampling with h_2 ($h_2 > h_1$)
Compute prediction errors $\{\epsilon(t_k)\}$ of $\hat{A}$ for the new samples
Perform t-test with null hypothesis: $\{\epsilon(t_k)\}$ comes from a normal distribution with zero mean.

Figure 4.7. A chart flow of the test criterion on system aliasing.

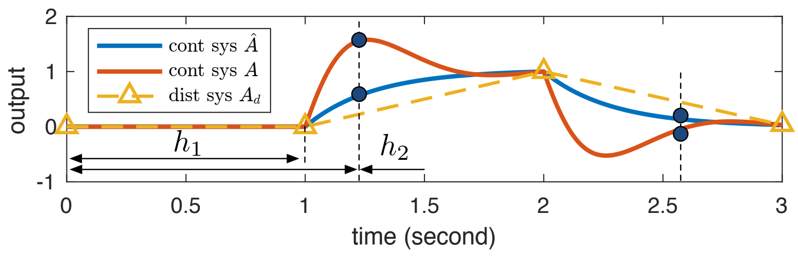


Figure 4.8. System responses of the dynamical system (4.1) with $A, \hat{A}$, where $A_d = \exp(h_1 A), \hat{A} = \operatorname{Log}(A_d)/h_1$, and the input signal is the square wave with period $2h_1$. The dots in deep blue are samples with the sampling period h_2.

where ϑ denotes the parameters under estimation, which parameterizes A, B, R, m_0, R_0 (e.g. see Åström (1980)). With the assumption that $w(t), x(t_0)$ are jointly Gaussian, the negative logarithmic likelihood function is

$$\begin{aligned} L(\vartheta) = -2\log p(Y^N|\vartheta) = \textstyle\sum_{k=1}^{N} \log\det\Lambda(t_k,\vartheta) \\ + \textstyle\sum_{k=1}^{N} \epsilon^T(t_k,\vartheta)\Lambda^{-1}(t_k,\vartheta)\epsilon(t_k,\vartheta) + \text{const}, \end{aligned} \tag{4.10}$$

where $\epsilon(t_k,\vartheta) := y(t_k) - \hat{y}(t_k|t_{k-1},\vartheta)$, $\hat{y}(t_k|t_{k-1},\vartheta)$ denotes the conditional mean of $y(t_k)$, and $\Lambda(t_k,\vartheta)$ the corresponding covariance matrix. The optimal prediction of $y(t_k)$ (i.e. $\hat{y}(t_k), \Lambda(t_k)$) is obtained using Kalman filters (e.g. Åström (1980); Ljung and Wills (2010)),

$$\begin{aligned} \hat{y}(t_k|t_{k-1}) &= C\hat{x}(t_k|t_{k-1}) \\ \hat{x}(t_k|t_k) &= \hat{x}(t_k|t_{k-1}) + K(t_k)\epsilon(t_k) \\ \frac{\mathrm{d}}{\mathrm{d}t}\hat{x}(t|t_k) &= A\hat{x}(t|t_k) + Bu(t), \quad t_k \le t \le t_{k+1} \\ K(t_k) &= P(t_k|t_{k-1})C^T\Lambda^{-1}(t_k) \\ P(t_k|t_k) &= P(t_k|t_{k-1}) - K(t_k)CP(t_k|t_{k-1}) \\ \frac{\mathrm{d}}{\mathrm{d}t}P(t|t_k) &= AP(t|t_k) + P(t|t_k)A^T + R, \quad t_k \le t \le t_{k+1} \\ \Lambda(t_k) &= CP(t_k|t_{k-1})C^T, \end{aligned} \tag{4.11}$$

where the initial condition is $\hat{x}(t_1|t_0) = m_0$, $P(t_1|t_0) = R_0$. Considering the equidistant sampling and assuming the input is constant over the sampling periods, the matrix Λ and K

appears in (4.11) can be treated as constant matrices by using steady-state Kalman filtering (Åström, 1980, Sec. 3.6).

Now consider the full-state measurement case (i.e. $C = I$) and restrict the noise to process noise. The calculation of prediction $\hat{y}(t_k|t_{k-1})$ becomes particularly simple since K in (4.11) always equals the identity, which yields

$$\begin{aligned}\epsilon(t_k, \vartheta) &= y(t_k) - A_d y(t_{k-1}) - B_d u(t_{k-1}),\\ \Lambda(t_k, \vartheta) &= R_d,\end{aligned} \tag{4.12}$$

where A_d, B_d, R_d are defined via A, B, R in (4.4), (4.5). Here we resolve $p(y(t_1)|\vartheta)$ by using $p(y(t_1)|y(t_0), \vartheta)$, where $y(t_0)$ is treated to be the deterministic and hence is removed from the conditional variables, and takes the first sample as its value. This simplification is due to the fact that $K \equiv I, P(t_k|t_k) \equiv 0$ and the measurement of x_0 is available (using $y(t_0)$), which also leads to that the best estimate of the distribution of x_0 is nothing better than a delta function (even if including the probability assumption of x_0 in maximum likelihood, i.e. $p(y(t_1)|\vartheta) = p(y(t_1)|y(t_0), \vartheta)p(y(t_0)|\vartheta)$ and $p(y(t_0)|\vartheta)$ takes the Gaussian density with mean m_0 and covariance matrix R_0). Alternatively, the likelihood function (4.10) can be obtained directly by considering $p(Y^N|\vartheta)$ without using Kalman filtering. However, the above standard procedure values when we deal with general cases (i.e. $C \neq I$). Noticing the particular parameterization (Ljung, 1999, p. 92, 206), maximizing likelihood can be performed as follows[2]:

$$\hat{\theta} = \operatorname{argmin}_\theta \textstyle\sum_{k=1}^N \epsilon(t_k, \theta)^T \epsilon(t_k, \theta), \tag{4.13a}$$

$$\hat{R}_d = \tfrac{1}{N} \textstyle\sum_{k=1}^N \epsilon(t_k, \hat{\theta})\epsilon^T(t_k, \hat{\theta}), \tag{4.13b}$$

where θ is composed of A, B. Note that this is not generally true for maximizing likelihood, which should be solved by (Ljung, 1999, p. 219) in general for multivariable cases. To estimate θ, instead of minimizing the prediction error as (4.13a), we will impose the l_1-penalty to favor the sparse solution in network reconstruction. This is mainly due to the observations in Section 4.2.1: the consistency of ML may fail to present us with a correct network structure unless appropriate thresholds of zero for each row of A, B are selected (difficult in practice).

Remark 4.3. If we include measurement noise[3] or consider the output measurement case $C \neq I$, the prediction includes the Kalman filter gain K, which depends on A, R and the covariance of measurement noise. It deserves to be emphasized that, due to the possible large sampling periods h, the numerical tricks used in (Åström, 1980; Ljung and Wills, 2010) are no longer valid to compute the gradient of the prediction error. We have to analytically calculate

[2]Similar to (Ljung, 1999, p. 219), one instead firstly minimizes the cost function analytically with respect to ϑ for every fixed R. Due to the particular parameterization, the resultant optimization no longer depends on R.

[3]Assume that it is reasonable to determine the network by A, B similarly, even though we don't have network models well-defined from the state-space representations with measurement noise.

the gradient as far as possible until the numerical computation is no longer restricted by h. This problem becomes fairly complicated.

4.4.1 Cost function in matrix forms and the gradients

The reconstruction algorithm is supposed to infer a sparse network, i.e. A is sparse. Due to the nonlinear least-square cost function (4.13a), it no longer satisfies the setup of Sparse Bayesian Learning proposed by Tipping (2001). Here we enhance sparsity by heuristically imposing the l_1-norm of A as the penalty to the cost function as the first tentative treatment.

Considering the measurement signal Y^N, let

$$\begin{aligned}
X_+ &\triangleq [y(t_1),\ y(t_2),\ \ldots,\ y(t_N)],\\
X_- &\triangleq [y(t_0),\ y(t_1),\ \ldots,\ y(t_{N-1})],\\
U_- &\triangleq [u(t_0),\ u(t_1),\ \ldots,\ u(t_{N-1})],
\end{aligned}$$

where $X_+, X_- \in \mathbb{R}^{n\times N}$. The matrix form of the l_1-regularised PEM problem is formulated as

$$\underset{A}{\text{minimize}}\ \left\|X_+ - \exp(hA)X_- - \int_0^h \exp(sA)\,\mathrm{d}s\,BU_-\right\|_F^2 + \lambda\,\|A\|_1\,, \tag{4.14}$$

where $\lambda \in \mathbb{R}^+$ and $h \in \mathbb{R}^+$ is the fixed and known sampling period, and the l_1 norm $\|A\|_1 \coloneqq \sum_{i,j=1}^n |A_{ij}|$ (A_{ij} denotes the (i,j)-th element of A). To avoid dealing with tensors, we use the vectorized form of (4.14) as follows:

$$\begin{aligned}
\underset{A}{\text{minimize}}\ &\Big\|\mathrm{vec}(X_+) - (X_-^T \otimes I_n)\,\mathrm{vec}(\exp(hA))\\
&-(U_-^T B^T \otimes I_n)\int_0^h \mathrm{vec}(\exp(sA))\,\mathrm{d}s\Big\|_2^2 + \lambda\,\|\mathrm{vec}(A)\|_1\,,
\end{aligned} \tag{4.15}$$

where $\mathrm{vec}(X_+) \in \mathbb{R}^{nN}$, $\mathrm{vec}(\exp(Ah)) \in \mathbb{R}^{n^2}$, $(X_-^T \otimes I_n) \in \mathbb{R}^{nN\times n^2}$, and $\otimes$ is the *Kronecker product*.

The problem (4.14) is challenging in optimization by noticing that it is: non-convex due to matrix exponential; not globally Lipschitz; and non-differentiable. The intuitive idea here is to use the the Gauss-Newton framework, in which each iteration is to solve a constrained l_1-regularized linear least square problem. Let

$$r(A,B) \triangleq \mathrm{vec}(X_+) - (X_-^T \otimes I_n)\,\mathrm{vec}(\exp(hA)) - (U_-^T B^T \otimes I_n)\int_0^h \mathrm{vec}(\exp(sA))\,\mathrm{d}s, \tag{4.16}$$

$\phi(A,B) \coloneqq r(A,B)^T r(A,B)$, and $f(A,B) \triangleq \phi(A,B) + \lambda\|\,\mathrm{vec}(A))\|_1$, which is the objective function of (4.15). Then $\min_A \phi(A,B)$ denotes the problem (4.15) without l_1-penalisation.

The gradient of $\phi(A,B)$ w.r.t. A is $\nabla_A\phi = 2J_A^T(A,B)r(A,B)$, where

$$J_A = -h(X_-^T \otimes I_n)K(hA) - (U_-^T B^T \otimes I_n)\int_0^h sK(sA)\,\mathrm{d}s, \tag{4.17}$$

and $K(A)$ is defined in Theorem A.4. The function $sK(sA), s\in[0,h]$ is integrable by noticing $\|sK(sA)\| \leq h\exp(h\|A\|)$ ($\|\cdot\|$ denotes any matrix norm). The matrix function $K(hA)$ and the integration can be calculated numerically given A. To compute the gradient of $\phi(A,B)$ w.r.t. B, using the matrix identity $\operatorname{vec}(AXB) = (B^T \otimes A)\operatorname{vec}(X)$ again for the term of U in (4.14), it yields $(U_-^T B^T \otimes I_n)\int_0^h \operatorname{vec}(\exp(sA))\,\mathrm{d}s = (U_-^T \otimes \int_0^h \operatorname{vec}(\exp(sA))\,\mathrm{d}s)\operatorname{vec}(B)$. Then it follows that $\nabla_B\phi = 2J_B^T(A,B)r(A,B)$, where

$$J_B = U_-^T \otimes \int_0^h \operatorname{vec}(\exp(sA))\,\mathrm{d}s. \tag{4.18}$$

If we assume B is diagonal and $\nabla_B\phi$ is calculated w.r.t. each diagonal element of B, then $\nabla_B\phi = 2\left((I\otimes\mathbf{1})J_B\right)^T r(A,B)$, where I is an $n\times n$ identity matrix and $\mathbf{1}$ is an n^2-dimensional row vector of 1's. In a sum, the gradient of $\phi(A,B)$ is

$$\nabla\phi = \begin{bmatrix}\nabla_A\phi\\ \nabla_B\phi\end{bmatrix} = 2\begin{bmatrix}J_A^T\\ J_B^T\end{bmatrix} r(A,B) \triangleq 2J^T(A,B)r(A,B). \tag{4.19}$$

4.4.2 Case I: update A with fixed B

Concerning the task of network reconstruction, we would like to infer a sparse A from data. As a special case, we only update A by solving (4.14) with B fixed to be B_0. For simplicity, in this subsection, let $r(A,B_0) \triangleq r(A), J_A(A,B_0) \triangleq J(A), \phi(A,B_0) \triangleq \phi(A)$ and $f(A,B_0) \triangleq f(A)$.

A linear approximation of $r(A)$ in a neighbourhood of a given point A_c is $r(A_c) + J(A_c)\operatorname{vec}(A-A_c)$. One may then use this approximation and formulate a l_1-regularized linear least squares problem

$$\underset{A}{\text{minimize}} \;\; \|r(A_c) + J(A_c)\operatorname{vec}(A-A_c)\|_2^2 + \lambda\|\operatorname{vec}(A)\|_1, \tag{4.20}$$

which can be solved to obtain an approximate solution to (4.15). Resolving it in an iterative way amounts to a Gauss-Newton method. However, $\operatorname{vec}(A-A_c)$ is not necessary to be a *descent direction* of (4.15).

In the k-th iteration, to guarantee the step $p_k = \operatorname{vec}(A-A_k)$ being a descent direction of (4.15), the search direction p_k is instead computed from the following constrained optimization problem

$$(P_1)\begin{cases}\underset{p_k\in\mathbb{R}^{n^2}}{\text{minimize}} & \|r(A_k) + J(A_k)p_k\|_2^2 + \lambda\|\operatorname{vec}(A_k)+p_k\|_1,\\ \text{subject to} & \sup_{g\in\partial f(A_k)} g^T p_k \leq 0,\end{cases}$$

where $\partial f(A_k)$ denotes the subdifferential of $f(A)$ at A_k, defined as

$$\partial f(A_k) \coloneqq \{\nabla\phi(A_k) + \lambda z : z \in J_1 \times \cdots J_{n^2}\}, \tag{4.21}$$

$$J_i \coloneqq \begin{cases} [-1,1] & \text{if } \operatorname{vec}(A_k)_i = 0 \\ \{1\} & \text{if } \operatorname{vec}(A_k)_i > 0 \\ \{-1\} & \text{if } \operatorname{vec}(A_k)_i < 0 \end{cases}, \tag{4.22}$$

in which $\operatorname{vec}(A_k)_i$ denotes the i-th element of $\operatorname{vec}(A_k)$. One may have noticed that the constraint in the problem (P_1) is the definition of *descent direction* for f at A_k, except replacing < 0 with $\leq -\epsilon$ to guarantee the existence of minimum. The problem (P_1) is a convex optimization problem by noticing that $\sup_{g\in\partial f(A_k)} g^T p_k$ is a convex function, which is a *pointwise supremum* over an infinite set of a linear function (Boyd and Vandenberghe, 2004, chap. 3). To solve the problem (P_1), we need to explore the constraint and derive an equivalent form (see Appendix 4.4.4 for details), given as follows.

$$(P_1') \begin{cases} \underset{p_k\in\mathbb{R}^{n^2}}{\text{minimize}} & \|r(A_k) + J(A_k)p_k\|_2^2 + \lambda\|\operatorname{vec}(A_k) + p_k\|_1, \\ \text{subject to} & \bar{g}(A_k)^T p_k + \lambda\|W(A_k)p_k\|_1 \leq 0, \end{cases}$$

where

$$\begin{aligned} \bar{g}(A_k) &= \nabla\phi(A_k) + \lambda\operatorname{sgn}(A_k), \\ W(A_k) &= I - \operatorname{diag}(|\operatorname{sgn}(A_k)|), \end{aligned} \tag{4.23}$$

the identify matrix I is of a compatible dimension, $|\cdot|$ denotes the element-wise absolute value, $\operatorname{diag}(v)$ denotes the diagonal matrix built from vector v, and the sgn function for vectors and matrices is extended from the standard signum function for real numbers, defined as follows: when $x \in \mathbb{R}^n$, $\operatorname{sgn}(x)$ denotes a n-dimensional vector whose i-th element equals $\operatorname{sgn}(x_i)$; and when $X \in \mathbb{R}^{m\times n}$, $\operatorname{sgn}(X) \coloneqq \operatorname{sgn}(\operatorname{vec}(X))$. Now the problem (P') can be easily modeled using CVX in MATLAB and solved by standard optimization solvers (see Grant and Boyd (2015)).

The iterate is updated via

$$\operatorname{vec}(A_{k+1}) = \operatorname{vec}(A_k) + s_k p_k, \tag{4.24}$$

where the step length s_k is determined by *backtracking line search*. Let $f'(A_k; p_k)$ denote the *directional derivative* of f at A_k in the direction p_k, by subcalculus,

$$\begin{aligned} f'(A_k; p_k) &\coloneqq \sup\nolimits_{g\in\partial f(A_k)} g^T p_k \\ &= \bar{g}(A_k)^T p_k + \lambda\|W(A_k)p_k\|_1. \end{aligned} \tag{4.25}$$

Given $\alpha \in (0, 0.5), \beta \in (0, 1)$ and an initial value $s_k = 1$, the line search is to perform $s_k \leftarrow \beta s_k$ until

$$f(A_k + \operatorname{ivec}(s_k p_k)) \leq f(A_k) + \alpha s_k f'(A_k; p_k). \tag{4.26}$$

The whole iterative method for (4.15) is summarized in Algorithm 3. One has to note that this algorithm cannot guarantee that the iterate will converges to the stationary point. It is lucky that we have good initial values of A_0, B_0 to start with that is provided by the *subspace method* in system identification. Solving (4.14) is to search a sparse A in the neighborhood of A_0. Moreover, we have the following propositions to guarantee fair properties of this algorithm. See Appendix 4.4.4 for the proofs.

Proposition 4.9. *Let $\hat{f}(A_k, p_k)$ denote the objective function of (P_1) and p_k^* be its optimal point. If $p_k^* \neq 0$ and $\sup_{g \in \partial f(A_k)} g^T p_k^* = 0$, then $0 \in \partial f(A_k)$.*

Proposition 4.10. *Let $\hat{f}(A_k, p_k)$ and p_k^* be defined in Proposition 4.9. If $p_k^* = 0$ and $0 \in \operatorname{argmin}_{p_k} \hat{f}(A_k, p_k)$, then $0 \in \partial f(A_k)$.*

Proposition 4.9 guarantees that the step p_k^* from solving (P_1) will always be a descent direction of (4.15) (i.e. $\sup_{g \in \partial f(A_k)} g^T p_k^* < 0$) until either it reaches the stationary point or $\{p_k^*\}$ converges to zero. When $\{p_k^*\}$ approaches to zero, there are two cases: one is Proposition 4.10 which guarantees that it reaches the stationary point; the other is described as follows:

$$p_k^* = 0 \text{ is the unique optimal point of } (P_1) \text{ and } 0 \notin \operatorname{argmin}_{p_k} \hat{f}(A_k, p_k).$$

Regarding the second case, indeed, if there exists other optimal point $p_k^{**} \neq 0$ of (P_1), we instead consider p_k^{**} using Proposition 4.9. This is why we restrict $p_k^* = 0$ to be the unique optimum of (P_1). In the second case, $\{p_k^*\}$ converges to zero and the objective value $f(A_k)$ also converges. However, in theory, we fail to prove that the limit point of $\{A_k + \operatorname{ivec}(p_k^*)\}$ is a stationary point. In the sense of applications, it could have been fairly good since we are looking up a sparse solution in the neighborhood of a good estimate. And note that Proposition 4.9 and 4.10 are not true in general for non-convex non-differential optimization using Gaussian-Newton approach together with descent direction constraints. Their proofs depends on certain properties of matrix exponential.

Remark 4.4. The proposed method can be considered as a variant of the *damped Gauss-Newton* method. If the Jacobian matrix $J(A_k)$ does not have full column rank, one could adopt the *Levenberg-Marquardt* method and solve

$$\begin{aligned} &\underset{p_k \in \mathbb{R}^{n^2}}{\text{minimize}} \quad \|r(A_k) + J(A_k)p_k\|_2^2 + \mu_k \|p_k\|_2^2 + \lambda \|\operatorname{vec}(A_k) + p_k\|_1, \\ &\text{subject to } \sup_{g \in \partial f(A_k)} g^T p_k \leq -\epsilon. \end{aligned}$$

Algorithm 3 Modified Gauss-Newton for l_1-regularized nonlinear least square problems

```
1: given $\epsilon > 0$, tolerance $\delta > 0$; $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$
2: initialize $A_0, B_0$ by the subspace method or zeros.
3: repeat
4:     Calculate $r(A_k), J(A_k), \nabla\phi(A_k)$ using (4.16), (4.17).
5:     Calculate $\bar{g}(A_k), W(A_k)$ using (4.23).
6:     Compute the search direction $p_k$ by solving problem $(P')$.
7:     Compute the directional derivative $f'(A_k; p_k)$ using (4.25).
8:     Determine the step size $s_k$ by backtracking line search (4.26).
9:     Update the iterate $A_k$ by (4.24).
10: until $\|A_k - A_{k+1}\|_2 < \delta$
```

4.4.3 Case II: update both A and B

Considering the vectorized form (4.15), let θ denote the optimal variables, i.e. $\theta \triangleq [\text{vec}(A)^T \ \text{vec}(\text{diag}(B))^T]^T$, where $\text{diag}(B)$ denotes the diagonal elements of B. The other notations follow that $r(A, B) \triangleq r(\theta), J(A, B) \triangleq J(\theta), \phi(A, B) \triangleq \phi(\theta)$ and $f(A, B) \triangleq f(\theta)$. In the same way as Section 4.4.2, the approximated l_1-regularized linear least square problem with constraints is written as

$$(P_2)\begin{cases} \underset{p_k \in \mathbb{R}^{2n^2}}{\text{minimize}} & \|r(\theta_k) + J(\theta_k)p_k\|_2^2 + \lambda\|\text{vec}(A_k) + \Lambda p_k\|_1, \\ \text{subject to} & \sup_{g \in \partial f(\theta_k)} g^T p_k \le 0, \end{cases}$$

where $\Lambda = [I \ \ 0]$ is of dimension $n^2 \times 2n^2$, $\partial f(\theta_k)$ denotes the subdifferential of $f(\theta)$ at θ_k, defined as

$$\partial f(\theta_k) := \{\nabla\phi(\theta_k) + \lambda\Lambda^T z : z \in J_1 \times \cdots J_{n^2}\},$$

and $J_i, i = 1, \ldots, n^2$ are defined in (4.22). Equivalently, we solve the following convex-constrained convex problem to update A, B by $\theta_{k+1} = \theta_k + p_k$.

$$(P_2')\begin{cases} \underset{p_k \in \mathbb{R}^{2n^2}}{\text{minimize}} & \|r(\theta_k) + J(\theta_k)p_k\|_2^2 + \lambda\|\text{vec}(A_k) + \Lambda p_k\|_1, \\ \text{subject to} & \bar{g}^T p_k + \lambda\|W(A_k)\Lambda p_k\|_1 \le 0, \end{cases}$$

where $\bar{g} \triangleq \nabla\phi(\theta) + \lambda\Lambda^T \text{sgn}(A_k)$ and $W(A_k) = I - \text{diag}(|\text{sgn}(A_k)|)$. The backtracking line search is equipped in the same way as Section 4.4.2 and the algorithm trivially follows by modifying Algorithm 3.

4.4.4 More details on optimization

Equivalence of (P_1) and (P_2)

The following shows how to derive (P_1') from (P_1). Consider $g \in \partial f(A_k)$ in (P_1), which implies $g = \nabla\phi(A_k) + \lambda z = \nabla\phi(A_k) + \lambda\,\text{sgn}(A_k) + \lambda(z - \text{sgn}(A_k)) \triangleq \bar{g} + \lambda(z - \text{sgn}(A_k))$.

Recall that $\max_{s\in[-1,1]^n} s^T x = \|x\|_1$ where x is an n-dimensional vector. Then we have

$$\begin{aligned}\sup_{g\in\partial f(A_k)} g^T p_k &= \bar{g}^T p_k + \lambda \sup_{z\in J_1\times\cdots\times J_{n^2}} (z - \operatorname{sgn}(A_k))^T p_k \\ &= \bar{g}^T p_k + \lambda \sup_{\bar{z}\in[-1,1]^l} \bar{z}^T (W(A_k)p_k) \\ &= \bar{g}^T p_k + \lambda\|W(A_k)p_k\|_1,\end{aligned}$$

where $W(A_k) = I - \operatorname{diag}(|\operatorname{sgn}(A_k)|)$ and $l = n^2 - \operatorname{card}(\operatorname{sgn}(A_k))$.

Consider the optimization (P_2), we go through the same procedure and obtain

$$\sup_{g\in\partial f(\theta)} g^T p_k = \bar{g}^T p_k + \lambda\|W(A_k)\Lambda p_k\|_1,$$

where $\bar{g} \triangleq \nabla\phi(\theta) + \lambda\Lambda^T \operatorname{sgn}(A_k)$.

Proof of Proposition 4.9

Proof. Without loss of generality, suppose that there exists $p_k^* \neq 0$ and $0 \notin \arg\min_{p_k} \hat{f}(A_k, p_k)$ such that $\sup_{g\in\partial f(A_k)} g^T p_k^* = 0$ and $0 \notin \partial f(A_k)$. (Indeed, if $0 \in \arg\min_{p_k} \hat{f}(A_k, p_k)$, we apply Proposition 4.10 and obtain $0 \in \partial f(A_k)$.) It implies that $\sup_{g\in\partial f(A_k)} g^T \alpha p_k^* = 0$, $\forall\alpha \in (0, 1]$. Hence, $f(A_k) \leq f(A_k + \alpha p_k^*)$ for small enough α. Let $\phi(A_k), \hat{\phi}(A_k, p_k)$ be defined by

$$\begin{aligned}f(A_k) &= \phi(A_k) + \lambda\|\operatorname{vec}(A_k)\|_1, \\ \hat{f}(A_k, p_k) &= \hat{\phi}(A_k, p_k) + \lambda\|\operatorname{vec}(A_k) + p_k\|_1,\end{aligned}$$

and hence $\phi(A_k + \operatorname{ivec}(p_k)) = \hat{\phi}(A_k, p_k) + o(\|p_k\|)$ ($\|\cdot\|$ denotes any vector norm). For simplicity, without any ambiguity, we use $f(A_k + p_k)$ to represent $f(A_k + \operatorname{ivec}(p_k))$. Then we have $f(A_k + p_k) = \hat{f}(A_k, p_k) + o(\|p_k\|)$, which yields

$$\lim_{\alpha\downarrow 0} \frac{|f(A_k + \alpha p_k^*) - \hat{f}(A_k, \alpha p_k^*)|}{\alpha\|p_k^*\|} = 0. \tag{4.27}$$

Now let us calculate this limit in a different way. Noting that $\hat{f}(A_k, p_k^*) \leq \hat{f}(A_k, \alpha p_k^*)$ (since $p_k^* \in \arg\min_{p_k} \hat{f}(A_k, p_k)$) and $\hat{f}(A_k, p_k^*) < \hat{f}(A_k, 0) = f(A_k)$ (since $0 \notin \arg\min_{p_k} \hat{f}(A_k, p_k)$), we have

$$\hat{f}(A_k, \alpha p_k^*) - f(A_k) \geq \hat{f}(A_k, p_K^*) - f(A_k) \geq \delta > 0.$$

Moreover, since $f(A_k + \alpha p_k^*) - f(A_k) = \phi(A_k + \alpha p_k^*) - \phi(A_k) + \lambda(\|A_k + \alpha p_k^*\|_1 - \|A_k\|_1) = o(\alpha\|p_k^*\|) + O(\alpha\|p_k^*\|)$, there exists $M \in \mathbb{R}$ such that

$$\lim_{\alpha\downarrow 0} \frac{|f(A_k + \alpha p_k^*) - f(A_k)|}{\alpha\|p_k^*\|} \leq M.$$

Now we recalculate the limit in (4.27) as follows

$$\begin{aligned}
&\lim_{\alpha\downarrow 0}\frac{|f(A_k+\alpha p_k^*)-\hat{f}(A_k,\alpha p_k^*)|}{\alpha\|p_k^*\|}\\
&\geq \lim_{\alpha\downarrow 0}\left|\frac{|f(A_k+\alpha p_k^*)-f(A_k)|}{\alpha\|p_k^*\|}-\frac{\left|f(A_k)-\hat{f}(A_k,\alpha p_k^*)\right|}{\alpha\|p_k^*\|}\right|\\
&\geq \left|M-\lim_{\alpha\downarrow 0}\frac{\delta}{\alpha\|p_k^*\|}\right| = +\infty,
\end{aligned}$$

which contradicts with (4.27). □

Proof of Proposition 4.10

Proof. Since $p_k^* \in \operatorname{argmin}_{p_k} \hat{f}(A_k,p_k)$, it yields $0 \in \partial_p \hat{f}(A_k,p_k^*)$. Note that, in our discussion, A_k is always fixed, and thus $\partial_p \hat{f}(A_k,p_k)$ denotes the subgradient of $\hat{f}(A_k,\cdot)$ at p_k. Now let us write $\partial_p \hat{f}(A_k,p_k)$ explicitly

$$\partial_p \hat{f}(A_k,p_k) = \left\{\nabla\hat{\phi}(A_k,p_k)+\lambda\hat{z} : \hat{z}\in \hat{J}_1\times\cdots\hat{J}_{n^2}\right\},$$

where

$$\hat{J}_i = \begin{cases}[-1,1] & \text{if } \operatorname{vec}(A_k)_i+(p_k)_i = 0\\ \{1\} & \text{if } \operatorname{vec}(A_k)_i+(p_k)_i > 0\\ \{-1\} & \text{if } \operatorname{vec}(A_k)_i+(p_k)_i < 0\end{cases},$$

$(p_k)_i$ denotes the i-th element of p_k, and $\nabla\hat{\phi}(A_k,p_k) = 2J(A_k)^T\left(r(A_k)+J(A_k)p_k\right)$. Hence $\partial_p\hat{f}(A_k,0) = \partial f(A_k)$. Therefore, $0\in\partial_p\hat{f}(A_k,p_k^*)$ with $p_k^*=0$ implies $0\in\partial f(A_k)$. □

4.5 System Aliasing: Sparsity and Bounded Constraints

In the previous section we hinted that the conditions for no *system aliasing* follow as a consequence of bounded eigenvalues. In this section we follow this path and study the problem in the presence of *system aliases*.

Consider the case of system aliasing, i.e. h is NOT chosen small enough such that $\mathscr{E}(A,I,h,\mathscr{S}_A)=\{A\}$. In order to find out A among the aliases we need extra information, for instance, the properties of A known *a priori*. Here we assume that the ground truth A is the sparest solution in $\mathscr{E}(A,I,h,\mathscr{S}_\kappa)$ and $\kappa\in\mathbb{R}$ as an upper bound that has been prescribed. The set $\mathscr{S}_\kappa$ will be defined after giving Definition 4.11. A can be searched by the criterion

$$\underset{\hat{A}\in\mathscr{E}(A,I,h,\mathscr{S}_\kappa)}{\text{minimize}}\ \|\hat{A}\|_0. \tag{4.28}$$

Here is a niche that is the calculation of $\mathscr{E}(A, I, h, \mathscr{S}_\kappa)$ from data. By definition,

$$\mathscr{E}(A, I, h, \mathscr{S}_\kappa) = \{\tilde{A} \in \mathscr{S}_\kappa : \exp(h\tilde{A}) = A_d\}, \tag{4.29}$$

where $A_d = \exp(hA)$. Even if we know A_d has consistent estimation via PEM or ML, considering the observations in Section 4.2.1, we know the workflow, that is estimating A_d and then obtaining $\mathscr{E}$ by matrix logarithms, is not robust in the presence of noise. In this section, we assume A_d can be perfectly estimated and focus on studying the possibility of searching A in the set of system aliases $\mathscr{E}(A, I, h, \mathscr{S}(\kappa))$ using the prior information.

Definition 4.11 (Z-weighted norm). Let $h_Z(A) = Z^{-1}AZ$, where Z is the matrix defined in Theorem A.2. Then the norm is defined as $\|h_Z(\cdot)\|_F = \|\cdot\|_F \circ h_Z$.

To formulate $\mathscr{S}_\kappa$, we introduce this special norm of A, which is equivalent to the Frobenius norm up to a change of coordinates. The matrix Z is constant, which can be obtained by Jordan decomposition of A_d. One can observe that

$$\|h_Z(\hat{A})\|_F = \text{vec}(\hat{A})^T (Z^T \otimes Z^{-1})^T (Z^T \otimes Z^{-1}) \text{vec}(\hat{A})$$

is a proper $(Z^T \otimes Z^{-1})^T (Z^T \otimes Z^{-1})$-weighted vector norm in terms of $\text{vec}(\hat{A})$. Using $\|h_Z(\cdot)\|_F$ is on the one hand simplifying the analysis we conduct throughout this section, and on the other explicitly penalizes the imaginary part of the eigenvalues without "distorting" them through the transformation by Z.

Now we define $\mathscr{S}_\kappa$ using the norm $\|h_Z(\cdot)\|_F$. The basic idea is that one should exclude such A's whose imaginary parts of eigenvalues are too large, which implies their system response will show wild fluctuation, as illustrated in Figure. 4.9. That's why we need to

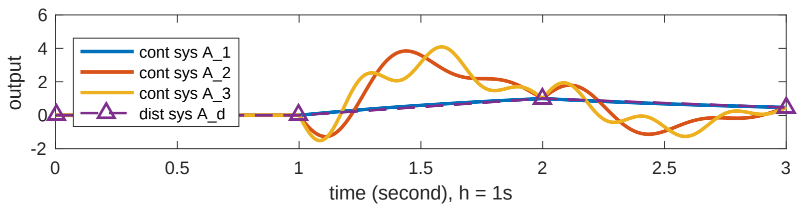


Figure 4.9. An example of system responses of multiple system aliases, i.e. A_i $(i=1,2,3)$ that satisfies $\exp(hA_i) = A_d$, where A_2 is the ground truth but A_1 is the principle logarithm, and the input signal is a square wave of period $2h$.

consider a reasonable set $\mathscr{S}_\kappa$ rather than $\mathbb{R}^{n\times n}$ in (4.29). In practice, even if we know the sampling frequency is not high enough to guarantee no system aliasing, we should still believe that the measurements do not miss too many fluctuation between samples. To make the constraint in (4.28) practically meaningful, we restrict $\mathscr{S}$ to be a norm bounded subset

$$\mathscr{S}_\kappa = \{\tilde{A} \in \mathbb{R}^{n\times n} : \|h_Z(\tilde{A})\|_F \leq \kappa\}. \tag{4.30}$$

In the following we will show that the feasible set of (4.28) has only finite elements, which implies it can be solved at least by brute force methods.

Let $M := \operatorname{diag}(m_1, m_2, \ldots, m_p)$, $\jmath := [j_1, j_2, \ldots, j_p]$[4]and $\beta := [\beta_1, \beta_2, \ldots, \beta_p]$, where $\log(\lambda_k) \triangleq \alpha_k + i\pi\beta_k$, $k = 1, \ldots, p$, and j_k, λ_k are defined in Theorem A.1. A function $\mathscr{I}$ is defined as

$$\mathscr{I}(\jmath, \delta) := \delta^T M \delta + (2\jmath + \beta)^T M \delta, \tag{4.31}$$

where $\jmath, \delta \in \mathbb{Z}^p$. Moreover, it satisfies $\mathscr{I}(\jmath,\delta) = \mathscr{I}(0, \jmath+\delta) - \mathscr{I}(0,\jmath)$, which follows by noticing

$$\mathscr{I}(\jmath,\delta) = (\delta + \jmath + \beta/2)^T M (\delta + \jmath + \beta/2) - (\jmath + \beta/2)^T M (\jmath + \beta/2). \tag{4.32}$$

Moreover, let A_0 denote a special matrix logarithm for which all j_k $(k = 1, \ldots, p)$ in (A.3) are equal to 0.

Definition 4.12 (equivalence relations). Let $\mathcal{S}$ denote the set of all primary matrix logarithms

$$\mathcal{S} := \{\tilde{A} \in \mathbb{R}^{n\times n} : \exp(h\tilde{A}) = A_d\}. \tag{4.33}$$

An *equivalence* relation "$\sim$" is defined on $\mathcal{S}$ as a binary relation: for any $A_1, A_2 \in \mathcal{S}$, $\jmath^{(1)}$ and $\jmath^{(2)}$ are defined for A_1, A_2, respectively, we say $A_1 \sim A_2$ if $\mathscr{I}(\jmath^{(1)}, \jmath^{(2)} - \jmath^{(1)}) = 0$.

Lemma 4.13. *Let $\mathcal{S}$ be the set defined in (4.29) and parametrized by (A.4) in Theorem A.2. For any $A_1, A_2 \in \mathcal{S}$, $\|h_Z(A_1)\|_F = \|h_Z(A_2)\|_F$ if and only if $A_1 \sim A_2$.*

Proof. Let $A_i := Z \operatorname{diag}(L_1^{j_1^{(i)}}, \cdots, L_p^{j_p^{(i)}}) Z^{-1}$, where $i = 1, 2$, $L_k^{j_k^{(i)}} := \log(J_k(\lambda_k)) + 2j_k^{(i)}\pi i I_{m_k}$, and all other notations are given in (A.4). Evaluate the difference

$$\begin{aligned}
\|h_Z(A_i)\|_F^2 - \|h_Z(A_0)\|_F^2 = {} & \operatorname{tr}\left(\operatorname{diag}^*(L_1^{j_1^{(i)}}, \cdots, L_p^{j_p^{(i)}}) \operatorname{diag}(L_1^{j_1^{(i)}}, \cdots, L_p^{j_p^{(i)}})\right) \\
& - \operatorname{tr}\left(\operatorname{diag}^*(L_1^{(0)}, \cdots, L_p^{(0)}) \operatorname{diag}(L_1^{(0)}, \cdots, L_p^{(0)})\right) \\
= {} & \textstyle\sum_{k=1}^p \operatorname{tr}\left(L_k^{j_k^{(i)}*} L_k^{j_k^{(i)}} - L_k^{(0)*} L_k^{(0)}\right) \\
= {} & \textstyle\sum_{k=1}^p \operatorname{tr}\left(2j_k^{(i)}\pi i \left(\log(J_k)^* - \log(J_k)\right) + 4\pi^2 j_k^{(i)2} I_{m_k}\right) \\
= {} & \textstyle\sum_{k=1}^p 4\pi j_k^{(i)} m_k (\beta_k + j_k^{(i)}) = 4\pi\, \mathscr{I}(0, \jmath), \quad \jmath \triangleq [j_1^{(i)}, \ldots, j_p^{(i)}],
\end{aligned} \tag{4.34}$$

[4]We use a different font type from j to avoid misunderstanding j as a scalar variable for indexes.

and it yields

$$\begin{aligned}\|h_Z(A_1)\|_F = \|h_Z(A_2)\|_F \Leftrightarrow \|h_Z(A_1)\|_F^2 - \|h_Z(A_0)\|_F^2 &= \|h_Z(A_2)\|_F^2 - \|h_Z(A_0)\|_F^2 \\ \Leftrightarrow \mathscr{I}(j^{(1)}, j^{(2)} - j^{(1)}) &= 0,\end{aligned}$$

which implies that $A_1 \sim A_2$ by definition. The first equality in (4.34) is due to the linear transformation $h_Z(\hat{A})$. □

Lemma 4.14. *Given any $\bar{A} \in \mathcal{S}$, there exists a finite number of $A_i \in \mathcal{S}$ that satisfies $A_i \sim \bar{A}$.*

Proof. Let j denote $[j_1, \dots, j_p]$ of $\bar{A}$ in (A.3), and $j^{(i)}$ denotes $[j_1^{(i)}, \dots, j_p^{(i)}]$ of $A_i \in \mathcal{S}$. $\delta \triangleq j^{(i)} - j$, therefore $\delta \in \mathbb{Z}^p$, where $\succeq$ denote the element-wise larger-or-equal relation. By Definition 4.12, it is equivalent to show that $\mathscr{I}(j, \delta) = 0$ has finite solutions, given j. We require δ to satisfy the following condition:

$$|\delta_i + j_i + \beta_i/2| \leq \sqrt{\frac{(j+\beta/2)^T M (j+\beta/2)}{m_i}} \tag{4.35}$$

for all $i = 1, \dots, p$. Otherwise, supposing that there exists $i \in \{1, \dots, p\}$ such that δ_i does not satisfy (4.35), we will have

$$\begin{aligned}\mathscr{I}(j,\delta) &= m_i(\delta_i + j_i + \beta_i/2)^2 + \sum_{k \neq i} m_k(\delta_k + j_k\beta_k/2)^2 \\ &- \sum_k m_k(j_k + \beta_k/2)^2 > \sum_{k\neq i} m_k(\delta_k + j_k\beta_k/2)^2 \geq 0.\end{aligned}$$

Let $\mathcal{S}' := \{A_i \in \mathcal{S} : j_k^{(i)} = \delta_k + j_k, \delta_k \text{ satisfies (4.35)}\}$. We have $\{A_i \in \mathcal{S} : A_i \sim \bar{A}\} \subseteq \mathcal{S}'$ and $\mathcal{S}'$ is a finite set. □

Lemma 4.15. *There exists a finite number of $A_i \in \mathcal{S}$ such that $\|h_Z(A_i)\|_F \leq \kappa$.*

Proof. Let $\kappa_0 \triangleq \|h_Z(A_0)\|_F$. Then we need to show there exists a finite number of $A_i \in \mathcal{S}$ such that $\|h_Z(A_i)\|_F^2 - \|h_Z(A_0)\|_F^2 \leq \kappa^2 - \kappa_0^2$, which is equivalent to show that there exists a finite number of solutions $\delta \in \mathbb{Z}$ to $\mathscr{I}(0,\delta) \leq (\kappa^2 - \kappa_0^2)/4\pi$. δ must satisfy the following condition:

$$|\delta_i + \beta_i/2| \leq \sqrt{\frac{(\beta/2)^T M (\beta/2) + (\kappa^2 - \kappa_0^2)}{m_i}} \tag{4.36}$$

for all $i = 1, \dots, p$. Otherwise, by supposing that there exists $i \in 1, \dots, p$ such that δ_i does not satisfy (4.36)leads to

$$\begin{aligned}\mathscr{I}(0,\delta) = m_i(\delta_i + \beta_i/2)^2 + \\ \sum_{k \neq i} m_k(\delta_k + \beta_k/2)^2 - (\beta/2)^T M(\beta/2) \\ > \sum_{k \neq i} m_k(\delta_k + \beta_k/2)^2 + (\kappa^2 - \kappa_0^2) \geq \kappa^2 - \kappa_0^2.\end{aligned}$$

Note that the set of all $\delta \in \mathbb{Z}$ that satisfies (4.36) is finite, which finalizes the proof. □

Proposition 4.16 (lower boundness of logarithms). *Let $\mathcal{S}$ be the set defined in (4.29). Given any $\bar{A} \in \mathcal{S}$, there exists $M(\bar{A}) > 0$, such that for any $A \in \{A \in \mathcal{S} : A \nsim \bar{A}\}$, it holds that*

$$\Big| \|h_Z(A)\|_F - \|h_Z(\bar{A})\|_F \Big| \geq M.$$

Proof. Let j denote $[j_1, \dots, j_p]$ of $\bar{A}$ in (A.3), N_{eqiv} be the number of A's that satisfy $A \sim \bar{A}$. Note that $|\|h_Z(A)\|_F^2 - \|h_Z(\bar{A})\|_F^2| = |(\|h_Z(A)\|_F^2 - \|h_Z(A_0)\|_F^2) - (\|h_Z(\bar{A})\|_F^2 - \|h_Z(A_0)\|_F^2)| = |\mathscr{I}(j,\delta)|, \delta \in \mathbb{Z}$, which implies it is equivalent to show that $|\mathscr{I}(j,\delta)|, \delta \in \mathbb{Z}$ has a non-zero lower bound if not considering the δ's that result in $\mathscr{I}(j,\delta) = 0$. We will prove it by contradiction. Assume this is not true, i.e. $\forall \epsilon > 0$ there exists δ such that $0 < |\mathscr{I}(j,\delta)| < \epsilon$. It implies that, arbitrarily given $\epsilon > 0$, there exists an infinite number of δ such that $\mathscr{I}(j,\delta) < \epsilon$, which is impossible since $\mathscr{I}(0, j+\delta) < \mathscr{I}(0,j) + \epsilon$ (using the fact that $\mathscr{I}(j,\delta) = \mathscr{I}(0, j+\delta) - \mathscr{I}(0,j)$) has a finite number of solutions provided by Lemma 4.15. □

Proposition 4.17. *Let $\mathcal{S}$ be the set defined in (4.29). For any $\bar{A} \in \mathcal{S}$, there exist $\kappa_l, \kappa_u \in \mathbb{R}$ in $\mathscr{S}(\kappa_l, \kappa_u) = \{\tilde{A} \in \mathbb{R}^{n \times n} : \kappa_l \leq \|h_Z(\tilde{A})\|_F \leq \kappa_u\}$ such that (4.28) has a unique optimal point in the sense of the equivalence relation in Definition 4.12.*

Proof. It immediately follows by choosing

$$\begin{aligned}\kappa_l &> \max\{0, \|h_Z(\bar{A})\|_F - M(\bar{A})\}, \\ \kappa_u &< \|h_Z(\bar{A})\|_F + M(\bar{A}),\end{aligned}$$

where $M(\bar{A})$ is the lower bound on the gap between $\bar{A}$ and any $A \nsim \bar{A} \in \mathcal{S}$, defined in Theorem 4.16. □

4.6 Numerical Examples

This section shows numerical examples of the proposed algorithm applied to 50 randomly generated datasets. The A matrices in state space models are chosen to be random stable sparse matrices. Data is sampled from the simulation of stochastic differential equations, with

$1/h$ set to be close to but larger than the critical sampling frequency by Theorem 4.5. The initial values of states were randomly sampled from Gaussian distributions with zero mean, and the process noise is Gaussian i.i.d, with SNR $= 0$ dB. Here, slightly abusing the name of "signals", this "SNR" value is defined as SNR $= 10\log(\sigma^2_{\text{init}}/\sigma^2_{\text{noise}})$, where σ^2_{init} denotes the variance of random initial states and σ^2_{noise} the variance of noise. Strictly speaking, SNR $= -\infty$ dB since the initial state is unknown in identification, and thus may not be treated as signals. We choose low sampling frequencies, large noise and limited samples to generate time series challenging in identification, however, which is a typical profile of time series in biological applications (e.g. microarray data He et al. (2012)).

The random generation of sparse stable A matrices is not a trivial task. Due to the lack of standards, it deserves time to explain our strategy to provide the information of randomness of A matrices. First, we do not want the network to be separable (i.e. a collection of separate small networks). Thus, we first generate a loop of 24 nodes, which is represented by a stable A matrix with nonzero diagonal and up-right (or bottom-left) corner elements. It serves as a base to build up A. Next a sparse matrix of the same dimension is generated with a fixed sparsity density. The A matrix is finally obtained by overlapping the sparse matrix and the base and then permuting rows and columns randomly. During the operation of overlapping two matrices, it might be possible that the combined matrix is no longer stable. Therefore, we need a test of stability before releasing A matrices. If the matrix turns into unstable, we simply discard it and search the next. The typical shapes of networks are shown as Figure 4.10.

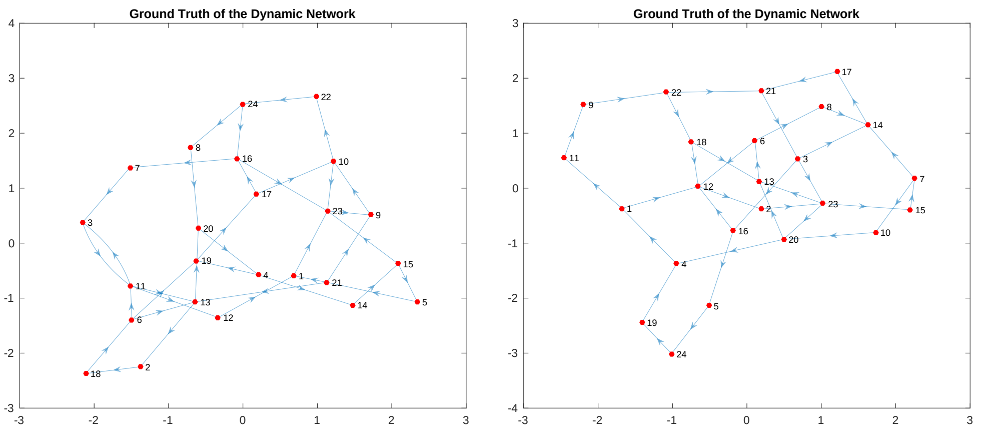

Ground Truth of the Dynamic Network
Ground Truth of the Dynamic Network

Figure 4.10. Examples of Boolean networks represented by the A matrices randomly generated.

An example of time series is given in Figure 4.11. The reconstruction results of this dataset is shown in Figure 4.12, together with the corresponding A_d's computed via matrix exponential. The straightforward way to estimate A is taking the principal matrix logarithm of PEM/ML solution $\hat{A}_d$, which is, however, contaminated by process noise and unable to give reasonable sparse structure of A, clearly shown as $\hat{A}_{\text{logm}}$in Figure 4.12a. Taking matrix

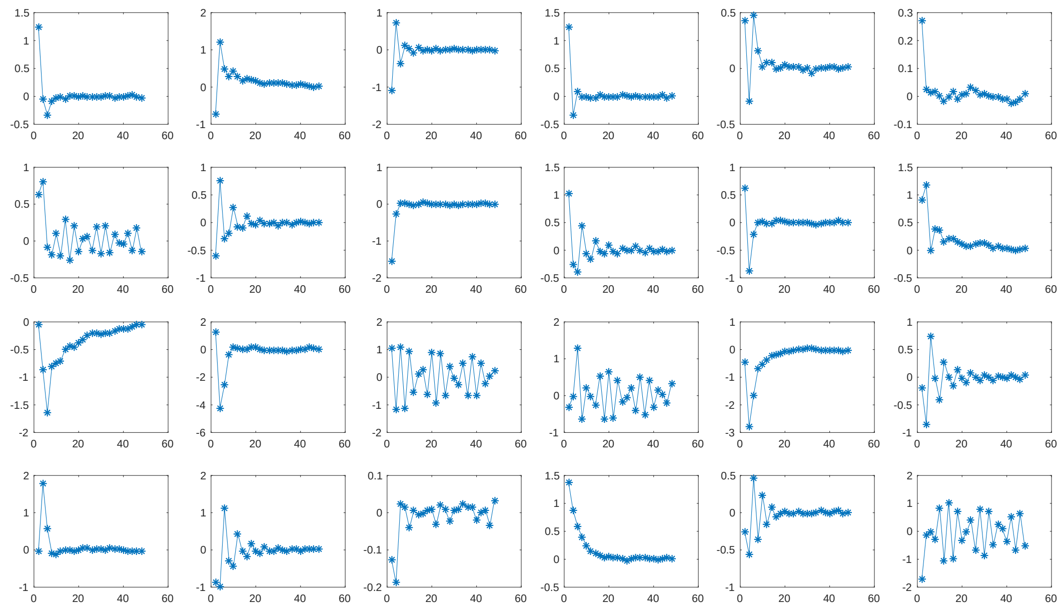
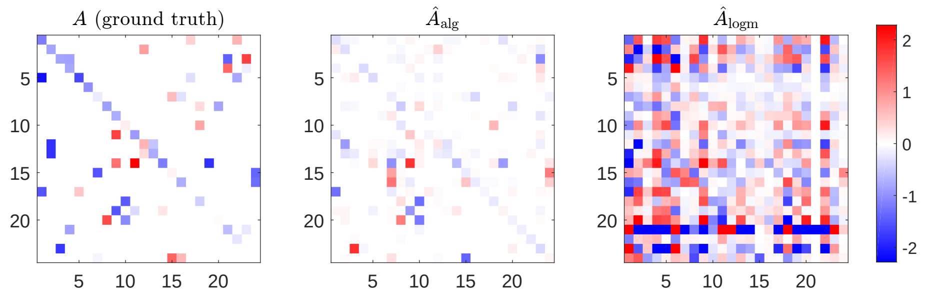


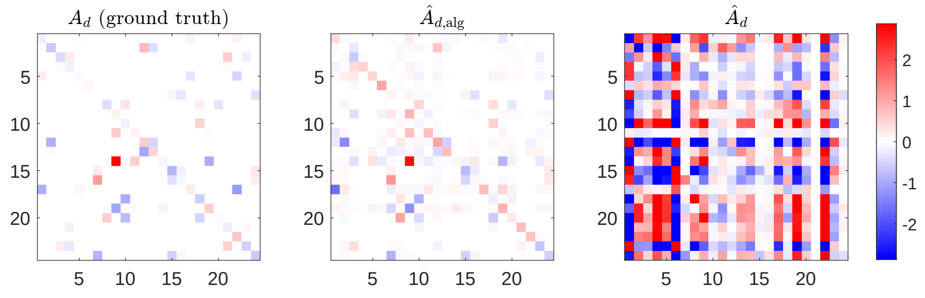
Figure 4.11. An example of time series used for reconstruction: 24 outputs, 24 samples, random initial states, no inputs, and SNR = 0 dB.

logarithm of least square estimations of A_d mostly encounters the issue of non-existence of principle logarithms, which results in complex values of A. This shows the effects of process noise on then estimation through matrix logarithms. However, the direct logarithm of $\hat{A}_d$ might also work well when the dimension is small (e.g. $\dim(A) \leq 6$). The curve of prediction errors is shown in Figure 4.13, which shows the convergence behaviors of Algorithm 3. Here the λ is chosen by performing network reconstruction on one dataset using λ logarithmically ranging from 10^{-4} to 100 and checking the sparsity of $\hat{A}$'s (users' prior knowledge) and the resultant prediction errors (whiteness, mean, standard deviations). This value of λ is then applied to all the other datasets. Indeed, the choice of λ also depends on A's, which, however, is randomly generated with the same sparsity. That explains why the same λ works almost well for all datasets. Alternatively, λ could be automatically calculated by running the cross-validation technique, when the amount of data allows. As widely used in bioinformatics, the Receiver Operating Characteristic (ROC) curve and the Precision-Recall (P-R) curve of this example are provided in Figure 4.14.

To show the performance of the proposed method, the ROC and the P-R curves, averaging over reconstruction results of 50 random systems, are shown in Figure 4.15. The variables used in ROC and P-R curves are computed by MATLAB function `perfcurve` with `XVals` fixed. However, one has to notice that, at certain values of "Recall" close to 0, the corresponding "Precision" is not defined, as shown in Figure 4.16a, due to the fixed `XVals`. However, we need to fixed the value of `XVals` in order to take average of 50 P-R curves. One may notice the irregular profile of P-R curves for certain datasets, where the corresponding values of "Precision" drop to zero in the neighborhood of zero "Recall".



(a) Ground truth and the estimations of A



(b) Corresponding A_d calculated by $\exp(hA)$

Figure 4.12. An example of network reconstruction results. A and A_d are the ground truth; $\hat{A}_{\mathrm{alg}}$ is estimated by the proposed method using $\lambda = 0.01$, and $\hat{A}_{d,\mathrm{alg}}$ is calculated by $\exp(h\hat{A}_{\mathrm{alg}})$; $\hat{A}_{\mathrm{logm}}$ is the principal logarithm of $\hat{A}_d$ divided by h, and $\hat{A}_d$ is estimated by PEM or ML.

The numerical examples are programmed and computed in MATLAB, and the codes will be released in public on the github.com/oracleyue. Considering computational efficiency, we directly used vector/matrix norms instead of quadratic forms in implementation (cf. (Grant and Boyd, 2015, chap. 11.1)). Thanks to the *matrix function toolbox* (Higham, 2008b) and the *CVX* (CVX Research, 2008) for easy usage of matrix functions and convex optimisation in MATLAB.

4.7 Conclusions

Continuous-time system identification is challenging when provided with low-sampling-frequency data of limited lengths. Unfortunately, this is a typical profile of time series in biomedicine applications. To reconstruct the correct dynamic networks from such time series, we have to identify the continuous-time models with sparse network structures. This paper studies the full-state measurement case, which is supposed to be the basic case while it shows particular complications. We first clarify the concept of system aliasing, which is raised by low sampling frequencies. A theorem on how to choose the sampling frequency to guarantee no system aliasing is provided, together with a test criterion. In regard to the "easy" case, i.e. no system aliasing, we present an algorithm to reconstruct sparse dynamic network from full-state measurements. In the case with system aliasing, the possibility on searching among

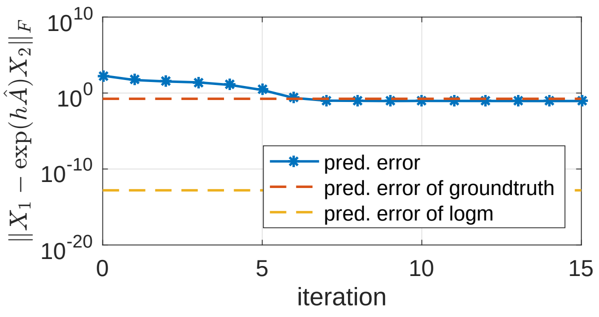
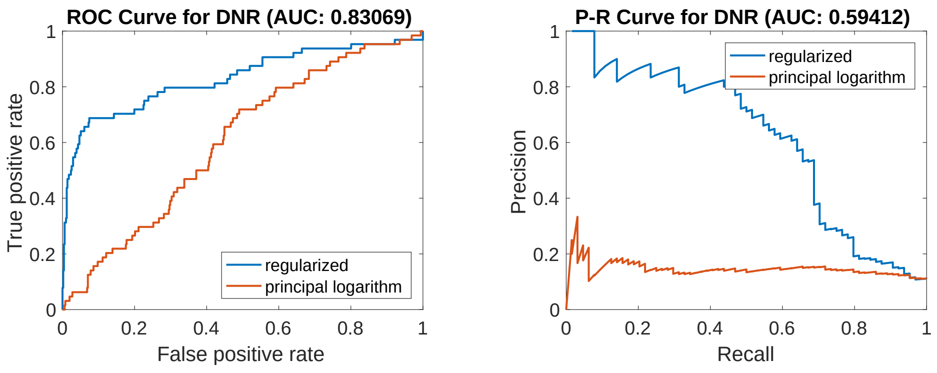


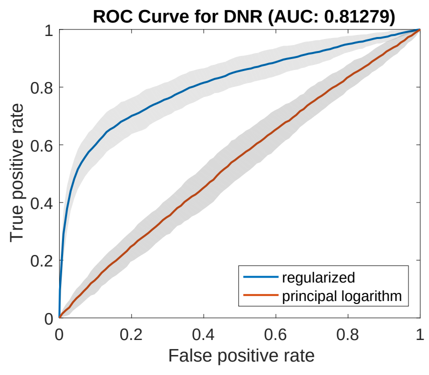
Figure 4.13. An example of convergence curve, with $\lambda = 0.01$ and zero as the initial point, for the chosen data set in Figure 4.11.



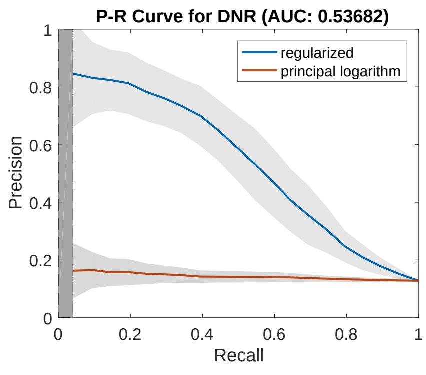
(a) ROC curve (b) P-R curve

Figure 4.14. The ROC and the Precision-Recall curves for the chosen example.

system aliases is manifested in theory relying on the prior information on network sparsity. With regard to the reconstruction performance, there remains czlconsiderable space to make an improvement.

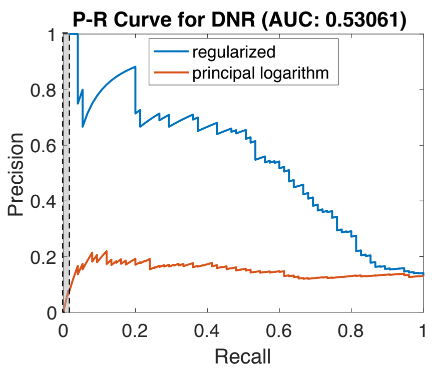


(a) ROC curve

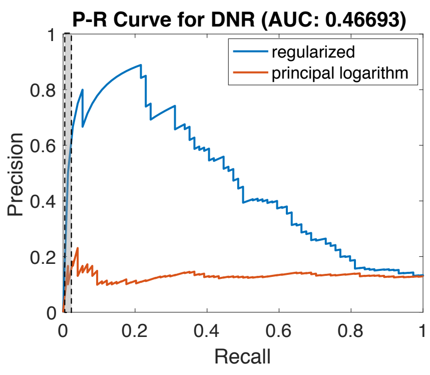


(b) P-R curve

Figure 4.15. The ROC and P-R curves averaging over the results of the proposed algorithm used on 50 random systems. The shaded area corresponds to one standard deviation from the mean value. The AUC values in titles refer to the mean of AUC values of the proposed method. The darkest area circled by dash lines is the uncertain region due to the average of `NaN` values in a few P-R curves.



(a) regular P-R curve



(b) irregular P-R curve

Figure 4.16. Examples of P-R curves which show regular shapes and irregular shapes. The shaded area in (a) is undefined region, which differs in different datasets; the one in (b) shows irregular profiles.

Chapter 5

Low Sampling Frequencies: Bayesian Methods

The importance of sampling frequencies in the network reconstruction has been shown in Chapter 4. Low sampling frequencies raise troubles in reconstruction due to the phenomena of system aliasing. In the cases of no system aliasing, we present reconstruction methods, which require full-state measurements. However, the demand on full-state measurements is too restrictive in practice. For example, in genetic regulatory networks, the interactions between genes are tuned or affected by intermediate RNAs or proteins, which are implausible to be measured in practice. This chapter considers the general low-sampling-frequency network reconstruction using output measurements. The EM method and Sparse Bayesian Learning (SBL) are used collaboratively to deal with issues due to low sampling frequencies in computation.

5.1 Problem Description

Consider a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,\infty)}, \mathbb{P})$, where the filtration is always assumed to be complete. Let $\{w(t) : t \geq 0\}$ be the n-dimensional standard $\mathcal{F}_t$-Brownian motion. The state-space representation of the dynamical system in continuous time is given by the following stochastic differential equation

$$\begin{aligned} \mathrm{d}x &= Ax\,\mathrm{d}t + Bu\,\mathrm{d}t + R\,\mathrm{d}w, \\ y &= Cx \end{aligned} \tag{5.1}$$

where $A \in \mathbb{R}^{n\times n}$ is stable, $B \in \mathbb{R}^{n\times n}$, $C = [I\ 0] \in \mathbb{R}^{p\times n}$, $R \in \mathbb{R}^{n\times n}$ is symmetric and semi-positive definitive, the initial $x(t_0)$ is a Gaussian random variable with mean m_0 and variance R_0, $t_0 \geq 0$, and $w(t)$ is interpreted as *disturbance* on the state variables (or called *process noise*). Since $C = [I\ 0]$, $x \triangleq [y^T\ z^T]^T$, where z is called *hidden states.* An input signal $\{u(t), t \geq t_0\}$ has been applied to the system and is assumed to be constant over the sampling periods. The output $y(t)$ of the system is sampled equidistantly at the time instants $t_0, t_1, ..., t_N$ with the sampling period h. For simplicity, in complicated formulas, we also use y_k to denote $y(t_k)$, similarly x_k, z_k for $x(t_k), z(t_k)$. The stochastic difference equation that relates the values of the state variable x in (5.1) at the sampling instants (Åström, 2012, p. 82-85); (Garnier and Wang, 2008, chap. 2), is given by

$$\begin{aligned} x(t_{k+1}) &= A_d x(t_k) + B_d u(t_k) + v(t_k), \\ y(t_k) &= C x(t_k), \end{aligned} \tag{5.2}$$

where

$$A_d = \exp(hA), \quad B_d = \int_0^h \exp(tA) B \, \mathrm{d}t, \tag{5.3}$$

$\exp(\cdot)$ is the matrix exponential, and the Gaussian i.i.d. $v(t)$ has mean zero and covariance matrix

$$R_d = \int_0^h \exp(tA) R \exp(tA^T) \, \mathrm{d}t. \tag{5.4}$$

The linear dynamic network model is given as

$$y(t) = Q(s)y(t) + P(s)u(t) + H(s)e(t), \tag{5.5}$$

where $Q(s), P(s)$ and $H(s)$ are $p \times p$, $p \times m$ and $p \times p$ matrices of strictly-proper real-rational transfer functions respectively, s is the differential operator[1] $sx(t) = \mathrm{d}x/\mathrm{d}t$, and $e(t)$ is the Gaussian white noise with zero mean and $\mathbb{E}[e(t)^T e(t')] = I\delta(t - t')$ (e.g. see Hayden et al. (2016b)). The model (5.5) is called *dynamical structure function* (DSF), first proposed by Goncalves and Warnick (2008). We can defines path diagrams from the network model to show the interconnections between the elements of the output variable.

Problem. Suppose that the underlying dynamic network $\mathcal{N}$ for the ground truth system (5.1) is sparse. Given the output measurements $Y^N \triangleq \{y(t_1), \ldots, y(t_N)\}$ sampled with a large sampling period h but without system aliasing, the task is to reconstruct sparse $\mathcal{N}$ from Y^N.

Remark 5.1. The network reconstruction problem is challenging in this setup due to the following reasons:

- The large sampling period h makes it particularly difficult to compute the gradient when estimating parameters using PEM (Prediction Error Minimization) or ML (Maximum Likelihood);

[1] In symbol conventions of signal processing, s is reserved for the frequency parameter in Laplace transform. Here we slightly abuse it to denote the differential operator in the time domain.

- It is difficult to impose sparsity constrains on $\mathcal{N}$, i.e. equivalently on $Q(q), P(q)$.
- To guarantee the network identifiability, additional constrains are required on $P(q)$ or A, B.

5.2 Prerequisites

5.2.1 Continuous-/Discrete-time models

Consider the underlying physical system (5.1), which derives the DSF (5.5) that shows the interconnections between the output variables. On the other hand, we have the discrete-time state-space representation (5.2), from which we can define the discrete-time DSF, denoted by $Q_d(z), P_d(z), H_d(z)$ (z is the shift operator, $zx(t) = x(t+1), z^{-1}x(t) = x(t-1)$), which describes the behavior of (5.5) evaluating at the sampling instants, i.e. for $k \in \mathbb{N}$,

$$y(t_k) = Q_d(z)y(t_k) + P_d(z)u(t_k) + H_d(z)e(t_k). \tag{5.6}$$

The discrete-time approach in network reconstruction is to determine $\mathcal{N}$ by identifying (5.6) from measurement signals. This method is valid when the sampling frequency is high enough, where the discrete-time DSF shares the same network topology with the continuous-time DSF. However, the property no longer holds when the sampling frequency is low. What's worse, we fail to have the direct map between $(Q(s), P(s), H(s))$ and $(Q_d(z), P_d(z), H_d(z))$ in theory without going through their state-space realizations. Thus, in the scenario of low sampling frequencies, the available methods based on identification of $(Q_d(z), P_d(z), H_d(z))$ barely help us. To identify $(Q(s), P(s), H(s))$ from low-sampling-frequency data, we need to identify the continuous-time state-space model first and then to derive the continuous-time DSF by definition, provided with the network identifiability. The idea is illustrated as Figure 5.1. Here the network identifiability becomes critical, since we need to guarantee that any state-space realization leads to the same DSF.

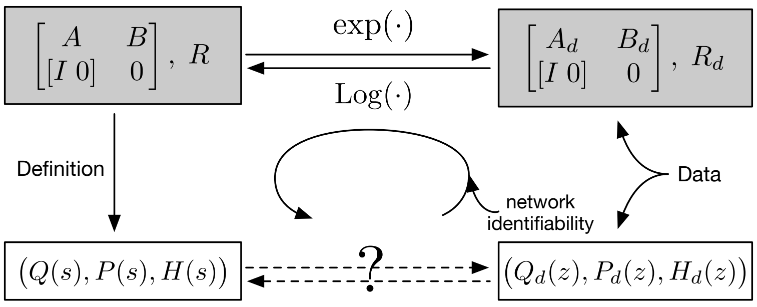

$\begin{bmatrix} A & B \\ [I\ 0] & 0 \end{bmatrix}, R$
exp(·)
Log(·)
$\begin{bmatrix} A_d & B_d \\ [I\ 0] & 0 \end{bmatrix}, R_d$
Definition
network identifiability
Data
$(Q(s), P(s), H(s))$
?
$(Q_d(z), P_d(z), H_d(z))$

Figure 5.1. The idea on identification of continuous-time DSFs with low sampling frequencies.

5.2.2 Network identifiability

There are two practical ways to guarantee the identifiability. One is to perturb each output variable by designed signals (Goncalves and Warnick, 2008), and, as a result, $Q(s)$ in the DSF is inferred using the signals responding to the input signals. The other is to taking advantages of noises (Hayden et al., 2016b). We need to accept *a priori* that the system is *minimum-phase* and the i.i.d. process noises perturb the outputs in such a way that $H(s)$ is square, diagonal and full-rank. And $Q(s)$ in the DSF is actually inferred from the responses of process noises. The benefit of the latter is the decrease of the number of experiments (i.e. $P(s)$ is no longer required to be square). However, in practice, it is hardly reasonable to accept these *a priori* assumptions on process noises. Therefore, in this chapter, we choose the first approach to guarantee network identifiability.

Referring to Theorem 2 in Goncalves and Warnick (2008), if $P(s)$ is square, diagonal and full-rank, the DSF $(Q(s), P(s), H(s))$ can be uniquely factorized from the transfer functions, and thereof the input-output data. To guarantee $P(s)$ to be diagonal, we need to impose constrains on (A, B) in the identification of state-space models. The constrains rely on the following proposition in Hayden et al. (2016b).

Proposition 5.1 (P-Diagonal Form 1 (Hayden et al., 2016b)). *Any DSF (Q, P) with P square, diagonal and full rank has a realization with A_{12}, A_{22}, B_1 and B_2 from* (2.3) *partitioned as follows:*

$$\left[\begin{array}{c|c} A_{12} & B_1 \\ \hline A_{22} & B_2 \end{array}\right] = \left[\begin{array}{cc|cc} \hat{c} & 0 & 0 & 0 \\ 0 & \times & 0 & B_{1_{22}} \\ \hline \hat{a} & \times & \hat{b} & 0 \\ 0 & \times & 0 & 0 \end{array}\right] \tag{5.7}$$

where $\times$ denotes an unspecified entry. The following is a canonical realization of $V = A_{12}(sI - A_{22})^{-1}B_2 + B_1$:

$$\left(\hat{a}, \quad [\hat{b} \ \ 0], \quad \begin{bmatrix} \hat{c} \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 \\ 0 & B_{1_{22}} \end{bmatrix}\right) \tag{5.8}$$

where $\hat{a} \coloneqq \mathrm{diag}(\alpha_1, \dots, \alpha_{p_{11}})$, $\hat{b} \coloneqq \mathrm{diag}(\beta_1, \dots, \beta_{p_{11}})$ and $\hat{c} \coloneqq \mathrm{diag}(\gamma_1, \dots, \gamma_{p_{11}})$, $p_{22} = \mathrm{dim}(B_{1_{22}})$, $p_{11} = p - p_{22}$ and where $(\alpha_i, \beta_i, \gamma_i, 0)$ is a minimal realization of $V(i, i)$ in controllable canonical form.

5.2.3 Likelihood and Kalman filtering

Consider the systems given by (5.1), whose output is sampled at the time instants $t_1, t_2, \dots, t_N$. The likelihood function is determined by the multiplication rule for conditional probability (Åström, 1980)

$$p(Y^N|\theta) \triangleq p(y(t_N),\dots,y(t_1)|\theta) = p(y(t_N)|y(t_{N-1}),\theta)\, p(y(t_{N-1})|y(t_{N-2}),\theta)\ \cdots p(y(t_1)|\theta),$$

where $p(y(t_k)|y(t_{k-1}),\theta)$ is the conditional p.d.f. (probability density function) of $y(t_k)$ given $y(t_{k-1}),\theta$; θ denotes the parameters under estimation, which parameterizes A, B, R, m_0, R_0. With the assumption that $w(t), x(t_0)$ are jointly Gaussian, the negative log likelihood function is

$$\begin{aligned} L(\theta) &= -2\log p(Y^N|\theta) \\ &= \textstyle\sum_{k=1}^N \log\det\Lambda(t_k,\theta) + \sum_{k=1}^N \epsilon^T(t_k,\theta)\Lambda^{-1}(t_k,\theta)\epsilon(t_k,\theta) + \text{const}, \end{aligned} \tag{5.9}$$

where $\epsilon(t_k,\theta) \coloneqq y(t_k) - \hat{y}(t_k|t_{k-1},\theta)$, $\hat{y}(t_k|t_{k-1},\theta)$ denotes the conditional mean of $y(t_k)$, and $\Lambda(t_k,\theta)$ the corresponding covariance matrix. The optimal prediction of $y(t_k)$ is obtained using Kalman filters (e.g., (Åström, 1980; Ljung and Wills, 2010)),

$$\hat{y}(t_k|t_{k-1}) = C\hat{x}(t_k|t_{k-1}) \tag{5.10a}$$

$$\hat{x}(t_k|t_k) = \hat{x}(t_k|t_{k-1}) + K(t_k)\epsilon(t_k) \tag{5.10b}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{x}(t|t_k) = A\hat{x}(t|t_k) + Bu(t), \quad t_k \le t \le t_{k+1} \tag{5.10c}$$

$$K(t_k) = P(t_k|t_{k-1})C^T\Lambda^{-1}(t_k) \tag{5.10d}$$

$$P(t_k|t_k) = P(t_k|t_{k-1}) - K(t_k)CP(t_k|t_{k-1}) \tag{5.10e}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P(t|t_k) = AP(t|t_k) + P(t|t_k)A^T + R, \quad t_k \le t \le t_{k+1} \tag{5.10f}$$

$$\Lambda(t_k) = CP(t_k|t_{k-1})C^T, \tag{5.10g}$$

where the initial condition is $\hat{x}(t_1|t_0) = m_0,\ P(t_1|t_0) = R_0$. Considering the equidistant sampling and assuming the input is constant over the sampling periods, the matrix Λ and K appears in (5.10) may be treated as constant matrices by using steady-state Kalman filtering (Åström, 1980, Sec. 3.6).

In the process of identification, provided with θ^k (the estimate of θ in the k-th iteration), the likelihood function or the cost function of PEM can be computed iteratively from t_0 to t_N by (5.10). To update the iterate θ^{k+1}, in (Åström, 1980; Ljung and Wills, 2010), the gradient of cost function in terms of θ is calculated using numerical tricks or formula simplifications, and then the parameter is updated via ML/MAP or PEM. However, due to the low sampling frequency, the available methods fail on calculating gradients. The analytic calculation of gradients is hardly possible due to no access to the higher derivative of matrix exponential functions and derivatives related to Kalman filters. Therefore, a new treatment deserves our efforts.

5.2.4 A glance on the idea

In the Expectation Maximization (EM) algorithm, the values of hidden states $z(t)$ at the sampling instants are considered as the "missing data". In the E-step, we compute the *expected complete data log likelihood* given Y^N. The M-step performs the MAP estimation, i.e. maximizing the expected complete data log likelihood plus the log prior, where a prior is used to impose sparsity (modified SBL) and the constrains are considered heuristically. In the M-step, the case reduces into network reconstruction using full-state measurements, which has been studied in Yue et al. (2016b) and will be solved instead by Bayesian approaches in this chapter. The whole idea is illustrated by Figure 5.2.

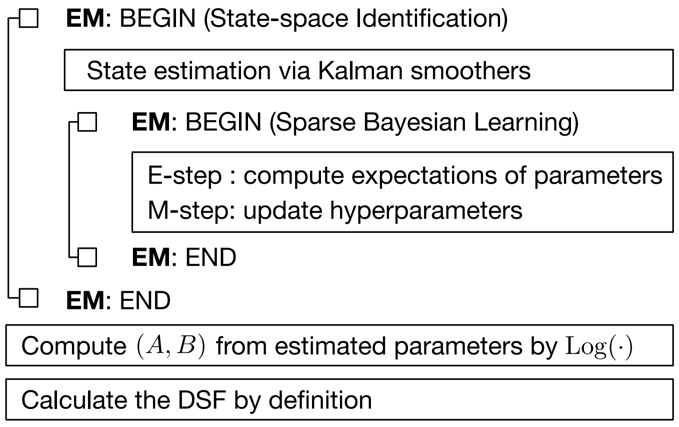

EM: BEGIN (State-space Identification)
State estimation via Kalman smoothers
EM: BEGIN (Sparse Bayesian Learning)
E-step : compute expectations of parameters
M-step: update hyperparameters
EM: END
EM: END
Compute (A, B) from estimated parameters by $\mathrm{Log}(\cdot)$
Calculate the DSF by definition

Figure 5.2. An overview of the idea on low-sampling-frequency network reconstruction.

5.3 Expectation Maximization

In the EM algorithm, we choose $z(t)$ as the latent variable, whose values at $t_1, \dots, t_N$ are the "missing data", denoted as $Z^N \triangleq \{z(t_1), \dots, z(t_N)\}$. The "complete data" is $X^N \triangleq \{x(t_1), \dots, x(t_N)\}$. Define the *complete data log likelihood* to be $\log p(X^N|\theta) = \log p(x(t_1)|\theta) + \sum_{k=2}^{N} \log p(x(t_k)|x(t_{k-1}), \theta)$, where $p(X^N|\theta)$ is the joint p.d.f. of $x(t_1), \dots, x(t_N)$ given θ; $p(x(t_k)|x(t_{k-1}), \theta)$ is the conditional p.d.f. of $x(t_k)$ given θ and the previous state value $x(t_{k-1})$. However, this cannot be computed directly due to the lack of the measurements of $z(t)$. Hence, the E-step is to calculate the *expected complete data log likelihood* using the available observations Y^N

$$\begin{aligned} \mathcal{Q}(\theta, \theta^k) &= \mathbb{E}_{\theta^k}\left(\log p(X^N|\theta)\,|Y^N\right) \\ &:= \int \log p(Y^N, Z^N|\theta)\, p(Z^N|Y^N, \theta^k) \mathrm{d}Z^N, \end{aligned} \tag{5.11}$$

where $\mathbb{E}_{\theta^k}(\cdot|Y^N)$ denotes the conditional expectation given the measurements Y^N and the parameters being θ^k (to be more clearly, the alternative notation is $\mathbb{E}_{Z^N|Y^N;\theta^k}(\cdot)$). Now we would like to introduce the prior distribution to impose sparsity of certain parameters that lead to sparse network structure. We categorize the parameters in θ into two groups: one is consist of unknown deterministic variables including m_0, R_0, R_d; the other comprises A, B, denoted by $\mathbf{w}$, that are random variables that commit a prior distribution. In the M-step, we perform MAP estimation to update θ:

$$\theta^{k+1} = \underset{\theta}{\operatorname{argmax}}\ \mathcal{Q}(\theta, \theta^k) + \log p(\mathbf{w}, \boldsymbol{\gamma}), \tag{5.12}$$

where $p(\mathbf{w}, \boldsymbol{\gamma})$ is the prior that depends on hyperparameter $\boldsymbol{\gamma}$. The specific construction of $p(\mathbf{w}, \boldsymbol{\gamma})$ is presented in Section 5.4. Note the prior in (5.12) might vary over iterates due to the hyperparameter that is determined by maximizing the *marginal likelihood function* (a.k.a. *evidence function*). It could be problematic to directly cite the usual result that the EM algorithm monotonically increases the log posterior of the observed data until it reaches a local optimum (see (Murphy, 2012, Sec. 11.4)), which uses a fixed prior distribution $p(\theta)$. The convergence deserves more discussions in further theoretical studies.

Now we explore how to calculate $\mathcal{Q}(\theta, \theta')$ for (5.1) in our study. In Lemma 5.2 and Lemma 5.3, to simplify notations, we use x_k to denote $x(t_k)$ and $\hat{x}_{k|l} \triangleq \hat{x}(t_k|t_l)$ that denotes the estimate of $x(t_k)$ given $x(t_l)$, similarly for y, z, u. The lemmas are similar to the ones proposed in Gibson and Ninness (2005), with modification due to the different choice of latent variables in the EM algorithm.

Lemma 5.2. *The expected complete data log likelihood $\mathcal{Q}(\theta, \theta')$ is given as follows (neglecting the constant terms)*

$$\begin{aligned} -2\mathcal{Q}(\theta, \theta') \quad &= \log\det R_0 + N \log\det R_d + \operatorname{Tr}\left\{R_0^{-1}\mathbb{E}_{\theta'}((x_0 - m_0)(x_0 - m_0)^T|Y^N)\right\} \\ &\quad + \textstyle\sum_{k=1}^N \operatorname{Tr}\left\{R_d^{-1}[\mathbb{E}_{\theta'}(x_k x_k^T|Y^N) - L\mathbb{E}_{\theta'}(x_k\xi_k^T|Y^N)^T\right. \\ &\qquad\qquad \left. - \mathbb{E}_{\theta'}(x_k\xi_k^T|Y^N)L^T + L\mathbb{E}_{\theta'}(\xi_k\xi_k^T|Y^N)L^T]\right\}, \end{aligned} \tag{5.13}$$

where $\xi_k \triangleq [x_{k-1}^T\ u_{k-1}^T]^T$, $L \triangleq [A_d\ B_d]$, and A_d, B_d, R_d are defined in (5.3), (5.4).

To compute (5.13), we need to calculate the following items: $\mathbb{E}_{\theta'}(x_k x_k^T|Y^N)$, $\mathbb{E}_{\theta'}(x_k x_{k-1}^T|Y^N)$, $\mathbb{E}_{\theta'}(x_k u_{k-1}^T|Y^N)$ and $\mathbb{E}_{\theta'}(x_{k-1}u_{k-1}^T|Y^N)$. This can be done via Kalman filters and smoothers. Note that the calculation of $\mathbb{E}_{\theta'}(u_{k-1}u_{k-1}^T|Y^N)$ is trivial, which is equal to $u_{k-1}u_{k-1}^T$ since $u(t)$ is assumed to be deterministic[2].

[2]In experiments we can be stochastic signals (e.g., the popular Gaussian i.i.d.) as $u(t)$ to stimulate systems. However, since $u(t)$ is assumed to be accessible in identification, it is still treated as deterministic.

Lemma 5.3. *Let the parameter vector θ' be composed of the elements of A, B, C, R, m_0, R_0, which define the system* (5.1)*. Then*

$$\mathbb{E}_{\theta'}(x_k x_k^T | Y^N) = \hat{x}_{k|N} \hat{x}_{k|N}^T + P_{k|N}, \tag{5.14}$$

$$\mathbb{E}_{\theta'}(x_k x_{k-1}^T | Y^N) = \hat{x}_{k|N} \hat{x}_{k-1|N}^T + M_{k|N}, \tag{5.15}$$

$$\mathbb{E}_{\theta'}(x_k u_{k-1}^T | Y^N) = \hat{x}_{k|N} u_{k-1}^T, \tag{5.16}$$

$$\mathbb{E}_{\theta'}(x_{k-1} u_{k-1}^T | Y^N) = \hat{x}_{k-1|N} u_{k-1}^T, \tag{5.17}$$

where $\hat{x}_{k|N}, P_{k|N}$ and $M_{k|N}$ are calculated in reverse-time recursions via the Kalman smoother

$$\begin{aligned} J_k &= P_{k|k} A_d^T P_{k+1|k}^{-1}, \\ \hat{x}_{k|N} &= \hat{x}_{k|k} + J_k(\hat{x}_{k+1|N} - \hat{x}_{k+1|k}), \\ P_{k|N} &= P_{k|k} + J_k(P_{k+1|N} - P_{k+1|k}) J_k^T, \end{aligned} \tag{5.18}$$

for $k = N, \ldots, 1$, and the lag-one covariance smoother

$$M_{k|N} = P_{k|k} J_{k-1}^T + J_k(M_{k+1|N} - A_d P_{k|k}) J_{k-1}^T \tag{5.19}$$

for $k = N, \ldots, 2$. The quantities $\hat{x}_{k|k}, P_{k|k}, P_{k|k-1}$ and initial conditions $\hat{x}_{N|N}, P_{N|N}$ required in (5.18), (5.19) *are computed by the Kalman filter* (5.10) *with replacing* (5.10c), (5.10f) *by*

$$\begin{aligned} \hat{x}_{k+1|k} &= A_d \hat{x}_{k|k} + B_d u_k, \\ P_{k+1|k} &= A_d P_{k|k} A_d^T + R_d, \end{aligned} \tag{5.20}$$

respectively, for $k = 1, \ldots, N$. The lag-one covariance smoother is initialized with

$$M_{N|N} = (I - K_N C) A_d P_{N-1|N-1}. \tag{5.21}$$

5.4 Sparse Bayesian Learning

This section is to construct an appropriate prior to impose sparse solutions. In regard to network sparsity, we focus on $Q(s)$. However, we cannot directly apply sparsity constraints on Q. Instead, we achieve it heuristically by imposing sparsity requirements on A. However, one deserves to be pointed out that it is possible that A is not sparse but Q is sparse, which case fails to be covered in this work.

In each iteration, let us treat x_0 as a fixed value so as to comply with the standard setup of SBL. (Strictly speaking, x_0 may be assumed to have the distribution $\mathcal{N}(m_0, R_0)$ and therefore should also be included in the likelihood $p(X^N|\theta)$, where m_0, R_0 are updated by Kalman smoothers in ML or MAP.) Hence, the complete-data likelihood function considered in this section is $p(X^N|\theta) \triangleq p(x_N, x_{N-1}, \dots, x_1|\theta) = p(x_N|x_{N-1}, \theta) \cdots p(x_2|x_1, \theta)p(x_1, \theta)$, where $p(x_k|x_{k-1}, \theta) = \mathcal{N}(x_k - A_d x_{k-1} - B_d u_{k-1}, R_d)$, $k = N, \dots, 1$. We rewrite the complete-data likelihood $p(X^N|\theta)$ as follows,

$$p(\mathbf{t}|\mathbf{w}; \Sigma) = (2\pi)^{-N_{\mathrm{t}}/2} |\Sigma|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{t} - \Phi\mathbf{w})^T \Sigma^{-1} (\mathbf{t} - \Phi\mathbf{w})\right], \tag{5.22}$$

where

$$\mathbf{t} = \begin{bmatrix} x_N \\ \vdots \\ x_1 \end{bmatrix}, \quad \Phi = \begin{bmatrix} \Phi_N \\ \vdots \\ \Phi_1 \end{bmatrix}, \; \Phi_k = \begin{bmatrix} x_{k-1}^T \otimes I & u_{k-1}^T \otimes I \end{bmatrix},$$

$$\mathbf{w} = \begin{bmatrix} \mathrm{vec}(A_d) \\ \mathrm{vec}(B_d) \end{bmatrix}, \quad \Sigma = \mathrm{blkdiag}(R_d, \dots, R_d) \; (N \text{ blocks}),$$

$N_{\mathbf{t}}$ denotes the dimension of $\mathbf{t}$, and here I denotes the $n \times n$ identity matrix. For simplicity, we also define the notation $\mathrm{ivec}(\mathbf{w}) \triangleq [A_d \; B_d]$. The parameter vector θ is composed of $\mathbf{w}$ and Σ, where $\mathbf{w}$ assumes a parametrized prior and Σ is treated as a deterministic parameter.

As studied in SBL, we introduce the Gaussian prior to impose sparsity on $\mathbf{w}$,

$$p(\mathbf{w}; \boldsymbol{\gamma}) = (2\pi)^{-N_{\mathrm{w}}/2} |K^{-1}\Gamma K^{-T}|^{-1/2} \exp\left(-\frac{1}{2}\mathbf{w}^T K^T \Gamma^{-1} K \mathbf{w}\right), \tag{5.23}$$

where $\Gamma = \mathrm{diag}(\boldsymbol{\gamma})$[3], $K = \mathrm{blkdiag}(K_{\mathrm{Log}}(I)/h, I)$, $K_{\mathrm{Log}}(I) = K_{\exp}^{-1}(0)$ and $K_{\exp}$ denotes the Kronecker form of the Fréchet derivative of matrix exponential, and N_{w} denotes the dimension of $\mathbf{w}$. The weight matrix K is used in the standard SBL prior to impose sparsity on A instead of A_d. The prior is constructed as follows. We expect the prior distribution of $\mathrm{vec}(A)$ is the zero-mean Gaussian with a diagonal variance matrix. In the prior, $\mathrm{vec}(A)$ is assumed to be in the neighborhood of 0. Hence, $\mathrm{vec}(A) = \mathrm{vec}(\mathrm{Log}(A_d)/h) \approx \mathrm{vec}(I)/h + K_{\mathrm{Log}}(I)(\mathrm{vec}(A_d) - I)/h$, which guides us to introduce the weight matrix K.

The posterior and marginal likelihood can be obtained in the procedure given by Tipping (2001). For fixed values of the hyperparameters, the complete-data posterior is Gaussian,

$$p(\mathbf{w}|\mathbf{t}; \boldsymbol{\gamma}, \Sigma) = (2\pi)^{-N_{\mathrm{w}}/2} |\Sigma_{\mathrm{w}}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu}_{\mathrm{w}})^T \Sigma_{\mathrm{w}}^{-1} (\mathbf{w} - \boldsymbol{\mu}_{\mathrm{w}})\right], \tag{5.24}$$

where

$$\begin{aligned} \boldsymbol{\mu}_{\mathrm{w}} &= \Sigma_{\mathrm{w}} \Phi^T \Sigma^{-1} \mathbf{t}, \\ \Sigma_{\mathrm{w}} &= \left(K^T \Gamma^{-1} K + \Phi^T \Sigma^{-1} \Phi\right)^{-1}. \end{aligned} \tag{5.25}$$

And the marginal likelihood is given by

$$p(\mathbf{t}; \boldsymbol{\gamma}, \Sigma) = (2\pi)^{-N_{\mathrm{t}}/2} |\Sigma_{\mathrm{t}}|^{-1/2} \exp\left(-\tfrac{1}{2}\mathbf{t}^T \Sigma_{\mathrm{t}}^{-1} \mathbf{t}\right), \tag{5.26}$$

[3]For convenience, we will use Γ and $\boldsymbol{\gamma}$ interchangeably when appropriate.

where $\Sigma_{\mathrm{t}} = \Sigma + \Phi K^{-1}\Gamma K^{-T}\Phi^{T}$. The hyperparameters $\boldsymbol{\gamma}$ and Σ will be determined by Evidence Maximization or Type-II Maximum Likelihood (Tipping, 2001).

Now the onus remains in estimating $\boldsymbol{\gamma}$ and Σ via evidence maximization of the complete-data marginal likelihood (5.26). We might use a rougher approximation that uses $\Sigma = \sigma^2 I$, where $\sigma \in \mathbb{R}_+$ and I denotes the identity matrix of the corresponding dimension, as the classical SBL (Tipping, 2001; Wipf and Rao, 2004, 2007). To determine $\boldsymbol{\gamma}$ and σ^2, we employ another EM algorithm to maximize $p(\mathbf{t};\boldsymbol{\gamma},\Sigma)$, which is equivalent to minimizing $-2\log p(\mathbf{t};\boldsymbol{\gamma},\Sigma)$. This EM algorithm proceeds by choosing $\mathbf{w}$ as the latent variable and minimizing

$$\mathbb{E}_{\mathbf{w}|\mathbf{t};\boldsymbol{\gamma},\Sigma}\left(-2\log p(\mathbf{t},\mathbf{w};\boldsymbol{\gamma},\Sigma)\right), \tag{5.27}$$

where $p(\mathbf{t},\mathbf{w};\boldsymbol{\gamma},\Sigma) = p(\mathbf{t}|\mathbf{w};\Sigma)p(\mathbf{w};\boldsymbol{\gamma})$. Instead of calculating (5.27) throughout, we calculate certain expectations in the E-step according to the demands of updating $\boldsymbol{\gamma}$ and σ^2 in the M-step. The evidence maximization is performed by computing the following the k-th iteration:

<u>E-step</u>:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{w}|\mathbf{t};\boldsymbol{\gamma}^k,\Sigma^k}(\mathbf{w}) = (\boldsymbol{\mu}_{\mathrm{w}})\big|_{\boldsymbol{\gamma}=\boldsymbol{\gamma}^k,\Sigma=\Sigma^k},\\
&\mathbb{E}_{\mathbf{w}|\mathbf{t};\boldsymbol{\gamma}^k,\Sigma^k}(\mathbf{w}\mathbf{w}^T) = (\Sigma_{\mathrm{w}} + \boldsymbol{\mu}_{\mathrm{w}}\boldsymbol{\mu}_{\mathrm{w}}^T)\big|_{\boldsymbol{\gamma}=\boldsymbol{\gamma}^k,\Sigma=\Sigma^k},
\end{aligned} \tag{5.28}$$

<u>M-step</u>:

$$\begin{aligned}
\Gamma^{k+1} &= \arg\min_{\boldsymbol{\gamma}} \mathbb{E}_{\mathbf{w}|\mathbf{t};\boldsymbol{\gamma}^k,\Sigma^k}\left(-2\log\hat{p}(\mathbf{t},\mathbf{w};\boldsymbol{\gamma},\Sigma^k)\right)\\
&= K\,\mathbb{E}_{\mathbf{w}|\mathbf{t};\boldsymbol{\gamma}^k,\Sigma^k}(\mathbf{w}\mathbf{w}^T)K^T,\\
(\sigma^2)^{k+1} &= \arg\min_{\boldsymbol{\gamma}} \mathbb{E}_{\mathbf{w}|\mathbf{t};\boldsymbol{\gamma}^k,\Sigma^k}\left(-2\log\hat{p}(\mathbf{t},\mathbf{w};\boldsymbol{\gamma}^k,\Sigma)\right)\\
&= \tfrac{1}{N}\left(\|\mathbf{t}-\Phi\boldsymbol{\mu}_{\mathbf{w}}\|_2^2 + (\sigma^2)^k\,\mathrm{Tr}(I-\Sigma_{\mathbf{w}}K^T(\Gamma^k)^{-1}K\right).
\end{aligned} \tag{5.29}$$

One has to note that we cannot directly compute $\boldsymbol{\mu}_{\mathbf{w}}$ and $\Sigma_{\mathbf{w}}$ by (5.25) as the classical SBL, since the complete data $x(t)$ is not accessible. What's worse, we have not yet known how to compute the expected value of $\boldsymbol{\mu}_{\mathbf{w}}$ and $\Sigma_{\mathbf{w}}$ analytically via Kalman filters or smoothers due the weighted matrices and the inverse[4]. Instead, we could approximately compute $\boldsymbol{\mu}_{\mathbf{w}}, \Sigma_{\mathbf{w}}$ using the state estimation from Kalman smoothers. which are used in (5.29). The details of the algorithm is manifested in Section 5.6. Here we have to admit that we fail to fully take advantage of information provided by Kalman filters and smoothers. The optimal variance estimation is discarded. The essential reason is that the conditional expectation of $\boldsymbol{\mu}_{\mathbf{w}}, \Sigma_{\mathbf{w}}$ depend on more than the first and second order moments of $x(t)$.

Without considerations on computational cost, there is a better alternative to compute the $\boldsymbol{\mu}_{\mathbf{w}}, \Sigma_{\mathbf{w}}$ using sampling methods. One way is to use particle filters to sample from

[4]Lemma 5.3 lists all the expected variables that can be computed from Kalman filter or smoother. Here the difficulty comes from the weighted matrix Σ^{-1}. One may obtain rough approximations by substituting the estimation of $x(t)$ from Kalman filter, which, however, does use the information of covariance provided by Kalman filters.

the distribution of the complete data $x(t)$, and then use the samples to estimate $\boldsymbol{\mu}_{\mathbf{w}}, \Sigma_{\mathbf{w}}$. Alternatively, in the special case of Gaussian $x(t)$, we could keep using Kalman filters and smoothers, which provide the optimal estimation of the mean and covariance of $x(t)$. Then we directly sample from the optimally estimated distribution of $x(t)$ and compute $\boldsymbol{\mu}_{\mathbf{w}}, \Sigma_{\mathbf{w}}$ in the Monte Carlo way.

5.5 Integration of Network Identifiability

To clarify how to to integrate identifiability constraints, we need to review the implementation of the SBL. The noise here is approximated by $\Sigma = \sigma^2 I$. Consider the noiseless environments, where we allow $\sigma^2 \to 0$. Using results from linear algebra, we have the following expression for $\boldsymbol{\mu}$ and $\Sigma_{\mathbf{w}}$:

$$\begin{aligned} \boldsymbol{\mu} &= \bar{\Gamma}^{1/2} \left(\Phi \bar{\Gamma}^{1/2}\right)^{\dagger} \mathbf{t} \\ \Sigma_{\mathbf{w}} &= \left[I - \bar{\Gamma}^{1/2} \left(\Phi \bar{\Gamma}^{1/2}\right)^{\dagger} \Phi \right] \bar{\Gamma}, \end{aligned} \tag{5.30}$$

where $\bar{\Gamma} = K^{-1} \Gamma K^{-T}$ and $(\cdot)^{\dagger}$ denotes the Moore-Penrose pseudoinverse. It is clear to see that the hyperparameters going to zero will lead to their corresponding $\mathbf{w}$ elements being zero. At the beginning of each EM step, we first prune the data matrices $\Phi, \mathbf{t}$ by checking elements in $\bar{\Gamma}$ that are smaller than the threshold of zero. To deal with noisy cases, the associated code by Wipf and Rao (2007) performs the SVD decomposition on $\Phi\bar{\Gamma}$, which is assumed to be USV^T and the estimation of $\boldsymbol{\mu}$ is updated by $\bar{\Gamma} V (S(S^2 + \hat{\sigma}^2 I + \epsilon I)^{-1}) U^T \mathbf{t}$, where $\hat{\sigma}^2$ is the estimated value in the previous step and ϵ is a fixed value close to zero, e.g., 10^{-16}. During EM iterations, the sizes of Φ and $\mathbf{t}$ will be significantly reduced (depending on the sparsity of $\mathbf{w}$), which guarantees the computational efficiency.

The objective of network identifiability integration is to guarantee the resultant P being diagonal. The key result used here is the *P-Diagonal Form* from Hayden et al. (2016b), as reviewed in Section 5.2.2. Let us start with the simple case, where each input perturbs each output node independently, i.e. $B = [\text{diag}(b_1, \ldots, b_p) \;\; 0]^T$. Instead of putting $\text{vec}(B_d)$ in $\mathbf{w}$, we only include $[b_1 \;\; \cdots \;\; b_p]$ and modify the data matrix Φ correspondingly. The alternative is to keep the form of $\mathbf{w}$ in (5.22), while in the implementation of the EM algorithm for SBL, we set the hyperparameters to zeros that correspond to zeros in B. The general case is more complicated due to the unknown dimension $p_{22} = \dim(B_{1_{22}})$ in Proposition 5.1. It is integrated in the same way that setting the hyperparameters to zeros that corresponds to zeros in (5.7) (note that $\hat{a}, \hat{b}, \hat{c}$ are diagonal with dimension $p - p_{22}$). Unfortunately, we have not fully understood the importance of p_{22}. It is possible that there exist multiple p_{22}'s leading to diagonal P's, which may or may not affect network reconstruction. It also has not yet been clear on how to select p_{22}, which "fortunately" has at most p choices (p_{22} is an integer). However, this issue becomes serious if the size of network (i.e. p) is huge.

5.6 Algorithm

In theory, all parameters are estimated in each M-step by optimizing (5.12). Nevertheless, due to the integration of SBL, it is not necessary to repeat solving the whole optimization problem. We can see that this optimization is separable. The parameters for initial states m_0, R_0 are treated as unknown deterministic variables, which can be estimated in each M-step by $m_0^k = \hat{x}_{0|N}, R_0^k = M_{0|N}$. This estimate comes from the study on optimizing $\mathcal{Q}(\theta, \theta')$ given in (5.13), e.g., see Gibson and Ninness (2005). Parameters R_d, A_d, B_d and hyperparameters $\boldsymbol{\gamma}$ are estimated via another inner-loop EM procedure for the SBL. The covariance matrix R_d is approximated as $\sigma^2 I$ and is estimated by (5.29) together with hyperparameters $\boldsymbol{\gamma}$. A_d and B_d are thought of as random variables, whose estimations are given by (5.25). Now we set up the EM framework with another embedded EM for SBL to present the whole algorithm, which is given as Algorithm 4.

Algorithm 4 SBL embedded in the EM framework

```
 1: Initialize $\mathbf{w}^0, \gamma^0, (\sigma^2)^0$ and $m_0^0, R_0^0$. Choose the dimension of state space $n$.
 2: while 1 do
 3:     Compute $\hat{x}_{k|N}, P_{x|N}, M_{x|N}$ and (5.14)-(5.17) $(k = 0, \dots, N)$ via Kalman filter and
        smoother in Lemma 5.3 using the given parameters $\mathbf{w}^k, \gamma^k, (\sigma^2)^k, m_0^k, R_0^k$.
 4:     Update $x_0^{k+1} = \hat{x}_{0|N}$, $R_0^{k+1} = M_{0|N}$.
 5:     Compute the Fréchet derivative $K$ using $\mathbf{w}^k$.
 6:     Compute $\Phi, \mathbf{t}$ using state estimations from Step 3.
 7:     Perform another EM procedure (5.28), (5.29) to obtain $\mathbf{w}^{k+1}$ and $(\sigma^2)^{k+1}$.
 8:     Update $\hat{A}_d, \hat{B}_d$ by $\mathbf{w}$, and $\hat{R}_d = \sigma^2 I$.
 9:     break if parameter estimate converges.
10: end while
11: Compute $\hat{A} = \mathrm{Log}(\hat{A}_d)/h$ and $\hat{B} = \left[\int_0^h \exp(tA)\mathrm{d}t\right]^{-1} B_d$.
12: Compute $Q(s), P(s)$ by definition.
```

5.7 Numerical Examples

The empirical study is performed on random stable sparse state-space models with $C = [I \quad 0], D = 0$, from with the sparse DSF models are derived. The way to generate such sparse stable state-space models is similar to Section 3.5.2. Here we do not include such sparse networks whose realizations' A matrices are not sparse, which cannot be tackled by the proposed method. The dimension of networks is $p = 40$, i.e., the dimension of output variables, and the dimension of states is set to $n = 100$. Considering the possibility that sparse state-space models lead to non-sparse networks, the randomly generated state-space models are reviewed by checking the sparsity of the corresponding DSF models.

The continuous-time state-space models are simulated, which will then be sampled with a specified sampling frequency to generate data for reconstruction. The choice of the sampling frequency is based on the studies of system aliasing, referring to Figure 4.1. Recall that the task in this study is to reconstruct networks of Case II in Figure 4.1. The Case I can also be treated by the proposed method, which can even be further simplified by only inferring discrete-time models since discrete-time and continuous-time DSF models share the same network topology. To generate problems of Case II, we first calculate the critical frequency of no system aliasing, and the sampling frequency is chosen to be close to (but larger than) this value.

To guarantee network identifiability from data, in simulation, p-variate Gaussian i.i.d. is used as input signals to drive each output node separately. It implies the case in consideration is $B = [I_{40\times 40} \;\; 0]^T$. The "general" case is not considered that inputs may drive outputs via hidden states but with guarantee that P is diagonal. It is due to the technical difficulty on generating B automatically for the "general" case due to randomness of A matrices. The "general" case is also barely useful in practice: one cannot know how to design the inputs that perturb nodes via hidden states without knowing the hidden state variables and models.

The proposed method runs on 100 random networks, where the dimension of states is set to 110 (the ground truth is 100). In the performance benchmark, we focus on Boolean structures of Q (P has been known to be diagonal and H is not interesting in applications). The reconstruction results are summarized in Table 5.1, where three columns of SNR show the averaged *true positive rates* (TPR, the percentage of correct links in results) and the column of "Failure" is the percentage of reconstruction results for 100 networks whose TPR are lower than 10%.

Table 5.1. Reconstruction results of discrete-time and continuous-time DSF models (rounded to zero decimals).

	SNR			Failure
	0 dB	20 dB	40 dB	
DSF(z)	70%	83%	94%	$<4\%$
DSF(s)	$<35\%$	56%	72%	24%

As shown in Table 5.1, the state-space based method (EM + SBL) provides a better way to perform reconstruction of discrete-time DSF's than choosing specific parametrization models (e.g., ARX, ARMAX, etc.). As explained in Chapter 3, we can only deal with ARX models due to difficulties on numerical optimization and such methods need to choose model orders of each element in Q, P. Here the DSF is derived from state space models, which allows more complicated time series models besides ARX. That is the main reason why the inference of discrete-time DSF models has better performance. Unfortunately, the performance of continuous-time DSF reconstruction is not really satisfactory. The main issue could be that the proposed method still relies on taking matrix logarithms ($\text{Log}(A_1)$ and $\text{Log}(A_2)$ could be significantly different even though sparse A_1, A_2 are quite close in the sense of matrix norms). This is obviously not an optimal solution even if we have modified the priors. The existence of "failure" cases is mainly due to the random construction of random networks, which cannot be covered by the proposed method but are difficult to be removed automatically in random model generation.

5.8 Conclusions

This chapter presents an algorithm to reconstruct networks from low-sampling-frequency data using EM algorithms and Sparse Bayesian Learning. Due to the lack of relation between continuous-time and discrete-time DSF models under low sampling frequencies, the algorithm reconstructs the continuous-time DSF's via identification of state-space models. Kalman filters and smoothers are used to provide state estimation. In order to guarantee network identifiability, which allows unique reconstruction of network structures, samples are measured from experiments that perturb each output independently. The identifiability conditions are integrated in the procedure of parameter estimation, so as to guarantee diagonal P from the inferred state-space model.

A few niches show the outlook of further studies. We did not discuss the method to choose the dimensions n of state spaces. Methods like AIC, BIC or cross-validation may help to determine this model complexity parameter. However, the results by the proposed method might not be sensitive to the choice of n considering its embedded sparse recovery, as long as n is close to the ground truth value and is large enough to model most dynamics related to hidden states. This question deserves more studies. The second issue is that the proposed method cannot deal with the case that the DSF model is sparse but its state-space realization is not. Better methods should be able to impose sparsity on DSF's directly.

5.9 Appendix

Proof of Lemma 5.2

Proof. Let us calculate the expected complete-data log likelihood $\mathbb{E}_{\theta'}\big(\log p(X^N|\theta)|Y^N\big)$. Noting that $(x_k|x_{k-1},\theta) \sim \mathcal{N}(A_d x_{k-1} + B_d u_{k-1}, R_d)$ for $k = 1, \ldots N$, we have $-2\log p(X^N|\theta) = \log\det R_0 + (x_0 - m_0)^T R_0^{-1}(x_0 - m_0) + N\log\det R_d + \sum_{k=1}^{N}(x_k - A_d x_{k-1} - B_d u_{k-1})^T R_d^{-1}(x_k - A_d x_{k-1} - B_d u_{k-1})$+const. Hence, $\mathbb{E}_{\theta'}(-2\log p(X^N|\theta)|Y^N) = \log\det R_0 + \mathbb{E}_{\theta'}\big((x_0 - m_0)^T R_0^{-1}(x_0 - m_0)|Y^N\big) + N\log\det R_d + \sum_{k=1}^{N}\mathbb{E}_{\theta'}\big((x_k - A_d x_{k-1} - B_d u_{k-1})^T R_d^{-1}(x_k - A_d x_{k-1} - B_d u_{k-1})|Y^N\big)$, where $\mathbb{E}_{\theta'}\big((x_0 - m_0)^T R_0^{-1}(x_0 - m_0)|Y^N\big) = \mathrm{Tr}\left\{R_0^{-1}\mathbb{E}_{\theta'}\big((x_0 - m_0)(x_0 - m_0)^T|Y^N\big)\right\}$ and

$$
\begin{aligned}
&\mathbb{E}_{\theta'}\Big((x_k - A_d x_{k-1} - B_d u_{k-1})^T R_d^{-1}(x_k - A_d x_{k-1} - B_d u_{k-1})|Y^N\Big)\\
&= \mathbb{E}_{\theta'}\Big(x_k^T R_d^{-1} x_k - x_k^T R_d^{-1} A_d x_{k-1} - x_k^T R_d^{-1} B_d u_{k-1} - x_{k-1}^T A_d^T R_d^{-1} x_k + x_{k-1}^T A_d^T R_d^{-1} A_d x_{k-1}\\
&\qquad + x_{k-1}^T A_d^T R_d^{-1} B_d u_{k-1} - u_{k-1}^T B_d^T R_d^{-1} x_k + u_{k-1}^T B_d^T R_d^{-1} A_d x_{k-1} + u_{k-1}^T B_d^T R_d^{-1} B_d u_{k-1}|Y^N\Big)\\
&= \operatorname{Tr}\Big\{ R_d^{-1}\mathbb{E}_{\theta'}(x_k x_k^T|Y^N) - R_d^{-1} A_d \mathbb{E}_{\theta'}(x_k x_{k-1}^T|Y^N) - R_d^{-1} B_d \mathbb{E}_{\theta'}(x_k u_{k-1}^T|Y^N)\\
&\qquad - A_d^T R_d^{-1}\mathbb{E}_{\theta'}(x_{k-1} x_k^T|Y^N) + A_d^T R_d^{-1} A_d \mathbb{E}_{\theta'}(x_{k-1} x_{k-1}^T|Y^N) + A_d^T R_d^{-1} B_d \mathbb{E}_{\theta'}(x_{k-1} u_{k-1}^T|Y^N)\\
&\qquad - B_d^T R_d^{-1}\mathbb{E}_{\theta'}(u_{k-1} x_k^T|Y^N) + B_d^T R_d^{1} A_d \mathbb{E}_{\theta'}(u_{k-1} x_{k-1}^T|Y^N) + B_d^T R_d^{-1} B_d \mathbb{E}_{\theta'}(u_{k-1} u_{k-1}^T|Y^N)\Big\}\\
&= \operatorname{Tr}\Big\{ R_d^{-1}\Big[\mathbb{E}_{\theta'}(x_k x_k^T|Y^N) - L\mathbb{E}_{\theta'}(x_k \xi_k^T|Y^N)^T - \mathbb{E}_{\theta'}(x_k \xi_k^T|Y^N)L^T + L\mathbb{E}_{\theta'}(\xi_k \xi_k^T|Y^N)L^T\Big]\Big\},
\end{aligned}
$$

with $\xi_k \triangleq \begin{bmatrix} x_{k-1} \\ u_{k-1} \end{bmatrix}$, $L \triangleq \begin{bmatrix} A_d & B_d \end{bmatrix}$. □

Proof of Lemma 5.3

Proof. The results follow straightforwardly by using Kalman Filters, Kalman Smoothers and lag-one covariance smoother (see reviews in Section 1.3.1). □

Chapter 6

Nonlinear Boolean Network Reconstruction

6.1 Introduction

In systems biomedicine, with time series available by many techniques (e.g., gene expression microarrays, RNA sequencing), it is expected that computational methods will help to identify the critical genes or pathways that are responsible for diseases. However, there are still many hurdles: large systems dimensions, nonlinearity, low sampling rates, etc. Many types of networks have been introduced in systems biology, e.g. *correlation networks*, *probabilistic Boolean networks*, *dynamic Bayesian networks*, etc. Nowadays, network reconstruction is expected to deliver causality information, rather than ambiguous relations in data.

Considering nonlinear dynamical systems, there is a lack of general network models similar to the DSF for LTI systems[1]. For the case of full-state measurements, i.e. when the entire state vector is measured, the definition of Boolean networks is straightforward: the directed edge $y_k \to y_1$ exists if y_k appears on the right-hand side of the differential equation of y_1, i.e. $\dot{y}_1(t) = f(..., y_k, ...)$. This perspective has been used in Pan et al. (2016, 2015), whose inference methods require full-state measurements and prior knowledge of dictionary functions. However, in the presence of *hidden states*, i.e. states that are not directly measured, the network model becomes particularly hard to define. This chapter is trying to give a useful definition of Boolean dynamic networks for nonlinear dynamical systems. The key motivation is to define Boolean dynamic networks without assumptions on *a priori* families of nonlinear basis functions, and to find reconstruction methods that no longer require measurements of all state variables.

[1]This concept may be understood by analogy to the DSF for LTI systems. The network here is required to provide direct causality. For instance, if keeping both direct and indirect causality, the causal network of mRNAs in the circadian clock model Pokhilko et al. (2010) would be a complete digraph, which is no longer useful.

This chapter generalizes the DSF to a large class of nonlinear dynamical systems allowing the existence of hidden states. The nonlinear systems in studies are of the form

$$\dot{x} = F(x) + Bu + Re \tag{6.1a}$$

$$y = [I \;\; 0]x \tag{6.1b}$$

where the state variable x is defined on M that is an open subset of $\mathbb{R}^n$, F is a smooth vector field defined on M, $B \in \mathbb{R}^{n\times m}$, $R \in \mathbb{R}^{n\times p}$, y is of dimension p ($p \leq n$), u is the control input of dimension m, and e is p-variate unknown white noise with covariance $\mathbb{E}[e(t)e^T(\tau)] = I\delta(t-\tau)$. Here, slightly abusing the notations, we use the model of white noise in engineering to simplify the DSF notation (2.2). To be strict, (6.1) should be described with respect to Brownian motion, and its meaning is assigned by stochastic integrals (Karatzas and Shreve, 2012; Mörters and Peres, 2010). Let $x = [y^T \; z^T]^T$ be the state variables in (6.1), where the elements in y are called the output variables and those in z are the hidden states (or *latent variables*). The later discussions are given in the language of differential manifolds, and thus the idea can be applied to the case where $F(x)$ is defined on an n-dimensional smooth manifold.

6.2 Nonlinear Boolean Dynamic Networks

6.2.1 Definition

Considering the dynamical system (6.1), where M is an open subset of $\mathbb{R}^n$ and $F : M \to M$ a smooth map, for each $x \in M$ we have the *differential of F at x* defined as a map, denoted by $dF_x : T_xM \to T_{F(x)}M$, where T_xM is the *tangent space to M at x* and $T_{F(x)}M$ the tangent space of M at the point $F(x)$. The matrix of dF_x in terms of the standard coordinate bases can be computed, denoted by A_x, which is irrelevant to the specific choice of coordinate charts. In particular, M is an open subset of Euclidean spaces, thus A_x is none other than the Jacobian matrix of F at x. Moreover, let TM denote the *tangent bundle of M*. The notations comply with Lee (2012) and one may consult (Lee, 2012, chap. 3) for more details.

Now, at $x \in M$, we have the dynamical system locally defined by the linear state-space representation (A_x, B, R). Following the standard definition of DSF for LTI systems (see Goncalves and Warnick (2008)), we have the dynamic network model defined locally as $(Q_x(q), P_x(q), H_x(q))$. Therefore, at $x \in M$, we have a Boolean dynamic network $\mathcal{G}_x^{\text{loc}}$ defined as a digraph from $(Q_x(q), P_x(q), H_x(q))$ (see Definition 2.1). Now the definition of nonlinear Boolean dynamic network naturally follows

Definition 6.1. A nonlinear Boolean dynamic network $\mathcal{G}$ of the nonlinear dynamic system (6.1) is defined as follows:

$$V(\mathcal{G}) := V(\mathcal{G}_x^{\text{loc}}),\ \forall x \in M, \quad E(\mathcal{G}) := \bigcup_{x\in M} E(\mathcal{G}_x^{\text{loc}}). \tag{6.2}$$

Alternatively, let us define the associated Boolean matrix C° of $C(q)$ as: $C_{ij}^\circ = 0$ if $C_{ij}(q) = 0$, otherwise 1. Then the nonlinear Boolean dynamic network $\mathcal{G}$ can be equivalently defined from $(Q^\circ, P^\circ, H^\circ)$, where $Q^\circ := \bigvee_x Q_x^\circ$, $P^\circ := \bigvee_x P_x^\circ$, $H^\circ := \bigvee_x H_x^\circ$, and $\bigvee$ denotes the element-wise logical disjunction operator.

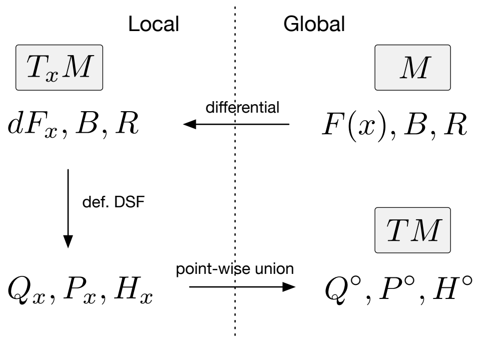


Figure 6.1. A diagram illustrating the procedure to give Definition 6.1 and the domains where each intermediate object of F, Q locates.

6.2.2 Issues on reconstruction

Definition 6.1 is based on M, which is forward invariant under a specific class of input signals. However, in the scenario of inference, only the outputs are accessible for the algorithm, which comprise but a subset of the state trajectories. To be precise, let $\mathcal{C}$ denote the signals of state variables when the dynamical system evolves through time, and $\mathcal{C} \subseteq M$. Let $M|_y := \{[I_{p\times p}\ 0]v : v \in M\}$, $\mathcal{C}|_y := \{y(t), t \geq 0\}$ and $\mathcal{C}|_y \subseteq M|_y$. The curve $\mathcal{C}|_y$ is the projection of $\mathcal{C}$ onto $M|_y$, as illustrated in Figure 6.1. The gap between M and $\mathcal{C}|_y$ is split into two parts: $M \Rightarrow \mathcal{C}$ and $\mathcal{C} \Rightarrow \mathcal{C}|_y$. The former one is resolved by a fundamental assumption, and the later is the network identifiability problem.

The gap between M and $\mathcal{C}$ (Fundamental Assumption)

Indeed, the systems can have complicated nonlinear dynamics, whose dynamical behaviors could depend on initial conditions, or the system structure may even change over time. However, it is not practically meaningful to consider that the underlying models of dynamical systems are varying due to the change of initial conditions. In fact, the selected nodes in

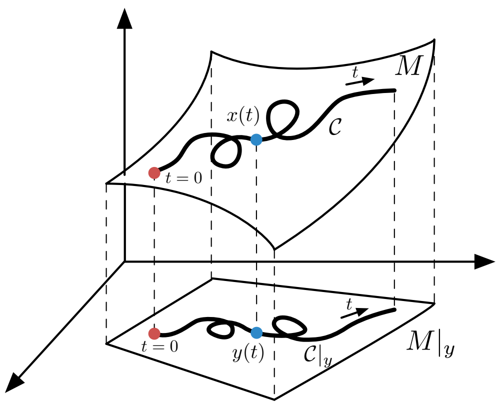


Figure 6.2. An illustration on the domains where state and output variables are defined; and the curves formed by their signals when the dynamical system evolves over time. Note that this illustrates high dimensional spaces, and there is no intersection of curves.

the applications usually have physical meanings. It is fair to assume that the structure of interactions between nodes is invariant to the changes of initial conditions.

Assumption 6.1. The interconnection structure between state variables in dynamical systems is independent from the initial conditions.

Let $\mathcal{G}|_{\mathcal{C}}$ denote the Boolean network defined on $\mathcal{C}$, i.e. $E(\mathcal{G}|_{\mathcal{C}}) = \bigcup_{x\in\mathcal{C}} E(\mathcal{G}_x^{\text{loc}})$. Assumption 6.1 implies that

$$E(\mathcal{G}) = E(\mathcal{G}|_{\mathcal{C}}).$$

Therefore, even though the total information M in Definition 6.1 cannot be accessed, we can obtain $\mathcal{G}$ by inferring the feasible $\mathcal{G}|_{\mathcal{C}}$, provided with known input signals.

The gap between $\mathcal{C}$ and $\mathcal{C}|_y$ (Network Identifiability)

A straightforward definition of identifiability of nonlinear Boolean dynamic network follows from Definition 2.8 and 6.1, which verifies linear network identifiability point-wisely. Section 6.2.1 tells that the nonlinear Boolean dynamic network is defined on $\{Q_x(q), P_x(q), H_x(q)\}_{x\in M}$, whose model structures can be denoted by $\{\mathcal{M}_x(\theta)\}_{x\in M}$. The ground truth value of θ may be different at each $x \in M$, which is denoted by $\theta^*(x)$.

Definition 6.2. Given the model structures $\mathcal{M}_x(\theta)$ at each point $x \in M$, the nonlinear Boolean dynamic network is globally (or locally) identifiable at $\theta^*(x)$ if, for any $x^* \in M$, each linear Boolean dynamic network with $\mathcal{M}_x(\theta)$ is globally (or locally) identifiable at $\theta^*(x^*)$.

This definition helps us to take advantage of well-known results on network identifiability. For example, if we apply ARX models for $x \in \mathcal{C}$ and each associated quadratic program has a unique solution, we know the nonlinear Boolean network is identifiable by definition.

6.3 Approximated Reconstruction Methods

According to Definition 6.1, the exact inference should be done by inferring $(Q_{x_t}, P_{x_t}, H_{x_t})$ by using the input-output measurement in a "small enough" neighborhood of x_t, and then taking union of their corresponding Boolean networks. However, this is not possible in practice due to two major issues: 1) no enough data for locally defined dynamic networks at each value of state variables; 2) not robust in the presence of noise. With regard to the first issue, we divide the whole time series into several segments, each of which is for approximated inference of a Boolean network at x_{t_k}. The second issue is resolved by adopting a technique dealing with heterogeneity in Yue et al. (2018) to guarantee the robustness by inferring all local networks $(Q_{x_t}, P_{x_t}, H_{x_t})$ together. However, this treatment leads to an approximated inference in general.

Consider the time series $\{y(t_i) : i \in \mathbb{N}\}$ in equidistant sampling. Suppose that we divide it into K segments $\{(y(t_{s_{k-1}}), y(t_{s_k})) : k = 1, \dots, K;\ s_{k-1}, s_k \in \mathbb{N}\}$, each of which corresponds to a neighborhood of $x^*_{t_k}$ (of which we may not know the value) and may not have the same length. Let $(Q_{x^*_{t_k}}, P_{x^*_{t_k}}, H_{x^*_{t_k}})$ denote the linear dynamic network locally defined for the segment $(y(t_{s_{k-1}}), y(t_{s_k}))$. For simplicity, we use (Q_k, P_k, H_k) instead. As presented in Yue et al. (2018), in the sense of Prediction Error Minimization (PEM), the inference of linear dynamic network can be formulated as p independent pseudo-linear regression problems, $\hat{y}_i(t|\theta_i) = \varphi^T(t, \theta_i)\theta_i$, where $\hat{y}_i$ is the one-step-ahead prediction of y_i, y_i the i-th element of p-dimensional output variable y, φ the regressor and θ_i the corresponding parameter vector. Note that once we get $\{\theta_1, \dots, \theta_p\}$, we obtain the network representation (Q, P, H). See Yue et al. (2018) for details on how the variables φ, θ_i are constructed from network model structures (i.e. the parametrization of DSF, e.g. ARX, ARMAX, etc.). In the following we will manifest how to integrate multiple linear network inference problems into one regression problems so as to improve robustness.

Without loss of generality[2], it is assumed to deal with the i-th output variable y_i. Letting

$$\mathbf{y}^{[k]} \triangleq \begin{bmatrix} y_i(t_{s_{k-1}}) \\ \vdots \\ y_i(t_{s_k}) \end{bmatrix}, \quad \hat{\mathbf{y}}^{[k]} \triangleq \begin{bmatrix} \hat{y}_i(t_{s_{k-1}}|\theta_i) \\ \vdots \\ \hat{y}_i(t_{s_k}|\theta_i) \end{bmatrix}, \quad \mathbf{A}^{[k]}(\mathbf{w}^{[k]}) \triangleq \begin{bmatrix} \varphi^T(t_{s_{k-1}}, \theta_i) \\ \vdots \\ \varphi^T(t_{s_k}, \theta_i) \end{bmatrix}, \quad \mathbf{w}^{[k]} \triangleq \theta_i,$$

[2]The whole network inference problem is solved by repeating the following operations on each element of the output variable y.

we have the formulation

$$\hat{\mathbf{y}}^{[k]} = \mathbf{A}^{[k]}(\mathbf{w}^{[k]})\, \mathbf{w}^{[k]}, \quad k = 1, \dots, K, \tag{6.3}$$

which can be rewritten into block matrices according to the "physical" meaning of each sub-vector of $\mathbf{w}^{[k]}$ (see Yue et al. (2017))

$$\begin{aligned} \mathbf{A}^{[k]} &\triangleq \begin{bmatrix} \mathbf{A}^{[k]}_{:,1} & \mathbf{A}^{[k]}_{:,2} & \cdots & \mathbf{A}^{[k]}_{:,N} \end{bmatrix}, \\ \mathbf{w}^{[k]} &\triangleq \Big[(\mathbf{w}^{[k]}_1)^T \quad \cdots \quad (\mathbf{w}^{[k]}_i)^T \quad \cdots \quad (\mathbf{w}^{[k]}_p)^T, \\ &\qquad (\mathbf{w}^{[k]}_{p+1})^T \quad \cdots \quad (\mathbf{w}^{[k]}_{p+i})^T \quad \cdots \quad (\mathbf{w}^{[k]}_{p+m})^T, \\ &\qquad (\mathbf{w}^{[k]}_N)^T \Big]^T, \end{aligned} \tag{6.4}$$

where p and m are the dimensions of output and input variables respectively, N the number of blocks. By optimizing $\min_{\mathbf{w}^{[k]}} \|\mathbf{y}^{[k]} - \hat{\mathbf{y}}^{[k]}\|$ (or including penalty of network sparsity), we get an estimate of $\mathbf{w}^{[k]}$ in the sense of PEM, which yields the linear dynamic network (Q_k, P_k, H_k) defined at $x^*_{t_k}$. If not considering the algorithmic robustness in the presence of noise, we solve the same problem for each segment and then take union of the results by definition. However, we would not recommend this treatment due to two major problems: 1) the results could be sensitive to the choice of segmentation (i.e. the number and the size of neighborhoods); 2) due to the deficiency of data for each neighborhood of x_t, the result may not be robust to noises.

Remark 6.1. To be precise, (6.3) should include one more constant term to further decrease the prediction errors

$$\hat{\mathbf{y}}^{[k]} = \mathbf{A}^{[k]}(\mathbf{w}^{[k]})\, \mathbf{w}^{[k]} + C^{[k]}\mathbf{1}, \quad k = 1, \dots, K,$$

where $C^{[k]}$ is a constant real number and $\mathbf{1}$ the vector of 1's of the same dimension as $\hat{\mathbf{y}}^{[k]}$. This constant term comes from the linearization of $F(x)$ in the neighborhood of $x^*_{t_k}$. Alternatively, if we keep using (6.3), we could update $\mathbf{y}^{[k]}$ with $\mathbf{y}^{[k]} - \bar{\mathbf{y}}^{[k]}$, in which $\bar{\mathbf{y}}^{[k]}$ denotes the mean of $\mathbf{y}^{[k]}$ multiplied by $\mathbf{1}$. Thus, for simplicity, we omit the constant term $C^{[k]}$ in equations.

Now consider improving the algorithmic robustness by inferring all local networks simultaneously to taking advantage of the whole dataset. Letting

$$\mathbf{w}_k \triangleq \begin{bmatrix} \mathbf{w}^{[1]}_k \\ \hline \vdots \\ \hline \mathbf{w}^{[K]}_k \end{bmatrix}, \quad \mathbf{w} \triangleq \begin{bmatrix} \mathbf{w}_1 \\ \hline \vdots \\ \hline \mathbf{w}_N \end{bmatrix},$$

we integrate all segments by stacking (6.3) of each segment and rearranging blocks of matrices, yielding (6.5), and use $\hat{\mathbf{y}} = \mathbf{A}(\mathbf{w})\mathbf{w}$ to denote (6.5b). Furthermore, we introduce two terms

$$\begin{bmatrix} \hat{\mathbf{y}}^{[1]} \\ \vdots \\ \hat{\mathbf{y}}^{[K]} \end{bmatrix} = \underbrace{\left[\begin{array}{ccc|c|ccc} \mathbf{A}^{[1]}_{:,1} & \dots & \mathbf{A}^{[1]}_{:,N} & & & & \\ & & & \ddots & & & \\ & & & & \mathbf{A}^{[K]}_{:,1} & \dots & \mathbf{A}^{[K]}_{:,N} \end{array}\right]}_{K\ \textbf{Blocks}} \left[\begin{array}{c} \mathbf{w}^{[1]} \\ \hline \vdots \\ \hline \mathbf{w}^{[K]} \end{array}\right] \tag{6.5a}$$

$$= \underbrace{\left[\begin{array}{ccc|c|ccc} \mathbf{A}^{[1]}_{:,1} & & & & \mathbf{A}^{[1]}_{:,N} & & \\ & \ddots & & \dots & & \ddots & \\ & & \mathbf{A}^{[K]}_{:,1} & & & & \mathbf{A}^{[K]}_{:,N} \end{array}\right]}_{N\ \textbf{Blocks}} \left[\begin{array}{c} \mathbf{w}_1 \\ \hline \vdots \\ \hline \mathbf{w}_N \end{array}\right] \tag{6.5b}$$

of group sparsity

$$\begin{aligned} \mathbf{w}^E &:= [\|\mathbf{w}_1^{[1]}\|_2, \cdots, \|\mathbf{w}_1^{[K]}\|_2 \cdots, \|\mathbf{w}_N^{[1]}\|_2, \cdots, \|\mathbf{w}_N^{[K]}\|_2]^T, \\ \mathbf{w}^S &:= [\|\mathbf{w}_1\|_2, \cdots, \|\mathbf{w}_N\|_2]^T, \end{aligned} \tag{6.6}$$

in which $\mathbf{w}^E \in \mathbb{R}^{KN}, \mathbf{w}^S \in \mathbb{R}^N$, $\|\cdot\|_2$ is the l_2-norm of vectors. One may have noticed that $\min_{\mathbf{w}} \|\mathbf{y} - \hat{\mathbf{y}}\|_2 + \lambda\|\mathbf{w}^E\|_0$ ($\lambda \in \mathbb{R}^+$ is the regularization parameter) is almost equivalent to infer sparse local dynamic networks (Q_k, P_k, H_k) $(k = 1, \dots, K)$ separately. Furthermore, as studied in Yue et al. (2018), the estimates via $\min_{\mathbf{w}} \|\mathbf{y} - \hat{\mathbf{y}}\|_2 + \lambda\|\mathbf{w}^S\|_0$ guarantees both network sparsity and the consistent network topology for all datasets (i.e. K segments), where the latter means $\mathcal{G}(Q_k, P_k, H_k) \equiv \mathcal{G}^0, \forall k$, in which $\mathcal{G}(Q_k, P_k, H_k)$ denotes the Boolean network determined from (Q_k, P_k, H_k) and $\mathcal{G}^0$ denotes any fixed Boolean network. However, Definition 6.1 implies (Q_k, P_k, H_k) could have the different network topology. Thus, we combined these two penalty terms to allow both multiple network topology and algorithmic robustness

$$\underset{\mathbf{w}}{\text{minimize}}\ \|\mathbf{y} - \mathbf{A}(\mathbf{w})\mathbf{w}\|_2^2 + \lambda_1\|\mathbf{w}^E\|_0 + \lambda_2\|\mathbf{w}^S\|_0, \tag{6.7}$$

where $\lambda_1, \lambda_2 \in \mathbb{R}^+$ are the regularization parameters. There is a trade-off between two penalty terms: $\lambda_1\|\mathbf{w}^E\|_0$ yields different sparse networks; $\lambda_2\|\mathbf{w}^S\|_0$ inclines (Q_k, P_k, H_k) $(k = 1, \dots, K)$ to have the same network topology. Assuming that the networks $(Q_k, P_k, H_k), k = 1, \dots, K$ share most arcs in common, the algorithmic robustness can be understood in the following sense: due to $\lambda_2\|\mathbf{w}^S\|_0$, the additional arcs of (Q_k, P_k, H_k) are included in the result only if it significantly contributes to decreasing the prediction error. This is also the reason why we emphasize that the presented approach is an approximated method. We increase the algorithmic robustness to noise by increasing the risk of missing arcs, which are not "significant" in the sense of predictability. In the language of Receiver Operating Characteristic (ROC) curve analysis, this implies a trade-off between the decrease of *false* *positive* (a false hit) and *false negative* (a miss). Here we stress that the decrease of *false positive* should be of higher priority in our consideration, whose large values make the inference results useless.

Remark 6.2. Definition 6.1 poses several interesting theoretical questions on the conditions which guarantee that the approximated method gives exact Boolean network reconstruction. It is possible that all local linear dynamic networks share the same network topology, which explains why and when Granger Causality applies successfully to nonlinear systems. Thus, it is particularly interesting to find out such conditions. Moreover, it is also likely that there exists only a finite number of different Boolean structures. Once we choose the segmentation correctly, the approximated method gives exact network reconstructions.

6.4 Algorithms

This section provides specific numerical algorithms to solve (6.7). We use the ARX parametrization as an example, in which $\mathbf{A}$ does not depend on $\mathbf{w}$ in (6.7). There are multiple way to heuristically solve convex-cardinality problems, e.g. l_1-norm heuristic approaches, *sparse Bayesian learning*. Here we give a solution using the classical l_1-heuristic treatment, which leads to the group LASSO problem (Yuan and Lin, 2006):

$$\underset{\mathbf{w}}{\text{minimize}} \, \|\mathbf{y} - \mathbf{A}\mathbf{w}\|_2^2 + \lambda_1 \|\mathbf{w}^E\|_1 + \lambda_2 \|\mathbf{w}^S\|_1, \tag{6.8}$$

where

$$\begin{aligned} \lambda_1 \|\mathbf{w}^E\|_1 &= \lambda_1 \textstyle\sum_{i=1}^N \sum_{j=1}^K \sqrt{\rho_{i,j}^E} \|\mathbf{w}_i^{[j]}\|_2, \\ \lambda_2 \|\mathbf{w}^S\|_1 &= \lambda_2 \textstyle\sum_{i=1}^N \sqrt{\rho_i^S} \|\mathbf{w}_i\|_2, \end{aligned} \tag{6.9}$$

$\rho_{i,j}^E$ denotes the dimension of $\mathbf{w}_i^{[j]}$, ρ_i^S the dimension of $\mathbf{w}_i$. To enhance sparsity further , one may use *Iterative Reweighted l_1 Method* (see Candes et al. (2008)), which is presented as follows

$$\begin{aligned} \mathbf{w}^{k+1} \leftarrow \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{A}\mathbf{w}\|_2^2 &+ \lambda_1 \textstyle\sum_{i=1}^N \sum_{j=1}^K \mu_{i,j}^k \sqrt{\rho_{i,j}^S} \|\mathbf{w}_i^{[j]}\|_2 \\ &+ \lambda_2 \textstyle\sum_{i=1}^N \nu_i^k \sqrt{\rho_i^S} \|\mathbf{w}_i\|_2, \end{aligned} \tag{6.10}$$

where

$$\begin{aligned} \mu_{i,j}^k &\leftarrow \left[\|(\mathbf{w}_i^{[j]})^k\|_2 + \epsilon^k \right]^{-1}, \\ \nu_i^k &\leftarrow \left[\|\mathbf{w}_i^k\|_2 + \epsilon^k \right]^{-1}, \end{aligned} \tag{6.11}$$

and k indexes the iterations, $\{\epsilon^k\}$ is a sequence converging to zero (see Candes et al. (2008)), e.g. $\epsilon^k \in (0, 1)$ is reduced by a factor of 10 until reaching the minimum of 10^{-10}.

Now we present algorithms using *Proximal Operators* and *ADMM* to solve large-dimensional (6.8) and (6.10). Let us consider (6.8) as

$$\underset{\mathbf{w}}{\text{minimize}}\ f(\mathbf{w}) + g(\mathbf{w}), \tag{6.12}$$

where $f(\mathbf{w}) \triangleq (1/2)\|\mathbf{y} - \mathbf{A}\mathbf{w}\|_2^2$, $g(\mathbf{w}) \triangleq \lambda_1\|\mathbf{w}^E\|_1 + \lambda_2\|\mathbf{w}^S\|_1$. Here we add 1/2 term in $f(\mathbf{w})$ to simplify calculations, which does not change the problem since the values of λ_1, λ_2 can be adjusted correspondingly. Given $\nabla f(\mathbf{w}) = \mathbf{A}^T(\mathbf{A}\mathbf{w} - \mathbf{y})$, the *Proximal Gradient Method* is to update $\mathbf{w}$ by $\mathbf{w}^{k+1} = \mathbf{prox}_{\gamma g}(\mathbf{w}^k - \gamma\nabla f(\mathbf{w}^k))$, $\gamma \in \mathbb{R}_+$, where k denotes the iteration index in proximal methods. Thus, the key step is to calculate the proximal operator $\mathbf{prox}_{\gamma g}(\mathbf{v})$. It is easy to see that $g(\mathbf{w}) = \sum_{i=1}^N g_i(\mathbf{w}_i)$, where $g_i(\mathbf{w}_i) := \lambda_1 \sum_{j=1}^K \sqrt{\rho_{i,j}^E}\|\mathbf{w}_i^{[j]}\|_2 + \lambda_2\sqrt{\rho_i^S}\|\mathbf{w}_i\|_2 \triangleq g_i^E(\mathbf{w}_i) + g_i^S(\mathbf{w}_i)$. Therefore, $\mathbf{prox}_{\gamma g}(\mathbf{v}) = \left[\mathbf{prox}_{\gamma g_1}(\mathbf{v}_1)^T \ \cdots \ \mathbf{prox}_{\gamma g_N}(\mathbf{v}_N)^T\right]^T$, where $\mathbf{v}$ is partitioned in the same way as $\mathbf{w}$ in terms of $\mathbf{w}_i, i = 1, \ldots, N$. However, it is difficult to calculate $\mathbf{prox}_{\gamma g_i}(\mathbf{v}_i)$ analytically. The *Dykstra-like Proximal Algorithm* in Combettes and Pesquet (2011) presents a numerical way to calculate this proximal operator, as shown in Algorithm 5.

Algorithm 5 Dykstra-like proximal algorithm for $\mathbf{prox}_{\gamma g_i}(\mathbf{v}_i)$

```
1: input: $\mathbf{v}_i$
2: output: $\mathbf{w}_i = \mathbf{prox}_{\gamma g_i}(\mathbf{v}_i)$
3: parameters: $\lambda_1, \lambda_2, \gamma, \rho_i^S, \rho_{i,j}^E, j = 1, \ldots, K$
4: given initial values: $\mathbf{w}_i^0 = \mathbf{v}_i$, $\mathbf{r}^0 = 0$, $\mathbf{z}^0 = 0$
5: repeat
6:     $\mathbf{s}^k \leftarrow \mathbf{prox}_{\gamma g_i^S}(\mathbf{w}_i^k + \mathbf{r}^k)$
7:     $\mathbf{r}^{k+1} \leftarrow \mathbf{w}_i^k + \mathbf{r}^k - \mathbf{s}^k$
8:     $\mathbf{w}_i^{k+1} \leftarrow \mathbf{prox}_{\gamma g_i^E}(\mathbf{s}^k + \mathbf{z}^k)$
9:     $\mathbf{z}^{k+1} \leftarrow \mathbf{s}^k + \mathbf{z}^k - \mathbf{w}_i^{k+1}$
10: until $\mathbf{w}_i$ converges
```

The proximal operators in line 6 & 8 in Algorithm 5 can be calculated by *block soft thresholding* (Parikh and Boyd, 2013), shown as follows

$$\begin{aligned}
\mathbf{prox}_{\gamma g_i^S}(\mathbf{x}) \ &= \left(1 - \gamma\lambda_2\sqrt{\rho_i^S}/\|\mathbf{x}\|_2\right)_+ \mathbf{x}\,, \\
\mathbf{prox}_{\gamma g_i^E}(\mathbf{x}) \ &= \left[\mathbf{prox}_{\gamma g_{ij}^E}(\mathbf{x}_1)^T \ \ldots \ \mathbf{prox}_{\gamma g_{ij}^E}(\mathbf{x}_K)^T\right]^T, \\
\mathbf{prox}_{\gamma g_{ij}^E}(\mathbf{x}_j) &= \left(1 - \gamma\lambda_1\sqrt{\rho_{ij}^E}/\|\mathbf{x}_j\|_2\right)_+ \mathbf{x}_j,
\end{aligned} \tag{6.13}$$

where $(\cdot)_+$ replaces each negative elements with 0, and $\mathbf{x}$ is partitioned in the same way as $\mathbf{w}_i$ in terms of $\mathbf{w}_i^{[j]}, j = 1, \ldots, K$. Now we have got $\mathbf{prox}_{\gamma g}(\mathbf{w})$ to run (*Accelerated*) *Proximal*

Gradient Method (see Yue et al. (2018) for similar implementations, or Parikh and Boyd (2013)).

To implement ADMM, the proximal operator of $f(\mathbf{w})$ needs to be calculated as

$$\mathbf{prox}_{\gamma f}(\mathbf{v}) = (I + \gamma \mathbf{A}^T\mathbf{A})^{-1}(\gamma \mathbf{A}^T \mathbf{y} + \mathbf{v}). \tag{6.14}$$

Given the proximal operators of $f(\mathbf{w})$ and $g(\mathbf{w})$, the ADMM algorithm is presented in Algorithm 6 (one may choose γ by standard line search methods for proximal gradient methods Parikh and Boyd (2013)). To solve the iterative reweighted l_1 problem (6.10), one

Algorithm 6 ADMM method

```
1:  Precompute $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{y}$
2:  given an initial value $\mathbf{w}^0, \mathbf{v}^0, \mathbf{u}^0$, $\gamma^0 = 1$, $\beta = 1/2$
3:  repeat
4:      Let $\gamma \leftarrow \gamma^k$
5:      repeat
6:          $\hat{\mathbf{w}} \leftarrow \mathbf{prox}_{\gamma f}(\mathbf{v}^k - \mathbf{u}^k)$ using (6.14)
7:          break if $f(\hat{\mathbf{w}}) \leq f(\mathbf{w}^k) + \nabla f(\mathbf{w}^k)^T(\hat{\mathbf{w}} - \mathbf{w}^k) + (1/2\gamma)\|\hat{\mathbf{w}} - \mathbf{w}^k\|_2^2$
8:          $\gamma \leftarrow \beta\gamma$
9:      until ;
10:     $\mathbf{w}^{k+1} \leftarrow \hat{\mathbf{w}}$, $\gamma^{k+1} \leftarrow \gamma$
11:     *Compute $\mathbf{prox}_{\gamma g_i}(\mathbf{w}_i^{k+1} + \mathbf{u}_i^k)$ using Algorithm 5 for all $i = 1, \ldots, N$
12:     $\mathbf{v}^{k+1} \leftarrow \mathbf{prox}_{\gamma g}(\mathbf{w}^{k+1} + \mathbf{u}^k)$
13:     $\mathbf{u}^{k+1} \leftarrow \mathbf{u}^k + \mathbf{w}^{k+1} - \mathbf{v}^{k+1}$
14: until any standard stopping criteria
                                    ▷ *Step 11 should be implemented in parallel if possible.
```

only needs to replace each $\sqrt{\rho_{i,j}^E}$ with $\mu_{i,j}\sqrt{\rho_{i,j}^E}$, and $\sqrt{\rho_i^S}$ with $\nu_i\sqrt{\rho_i^S}$ in the algorithms presented above.

6.5 Numerical Examples

This section uses the Millar-10 model as an example to perform the proposed method. The complete nonlinear ordinary differential equations have 19 state variables (mRNAs and proteins), of which 8 mRNAs and 1 protein are output variables (LHY, TOC1, Y, PPR9, PPR7, NI, GI, ZTL), which can be measured in biological experiments (see the supplement of Pokhilko et al. (2010) for details). The model simulates the behaviors of eukaryotic circadian clocks, whose outputs are sampled by 30 min, plotted in Figure 6.3. The light is controlled as follows: -48h∼0h is two cycles of light-dark, each of which lasts for 12 hours; 0h∼48h is always in light, illustrated as a square wave in Figure 6.3. The ground truth of interaction path diagram of key mRNAs for circadian clocks is illustrated in Figure 6.4a.

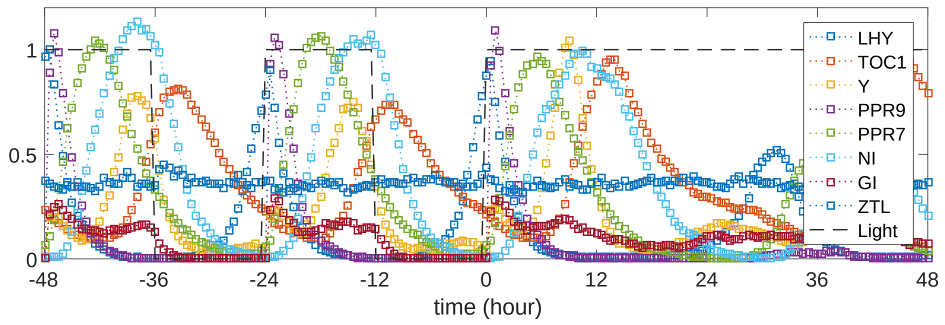


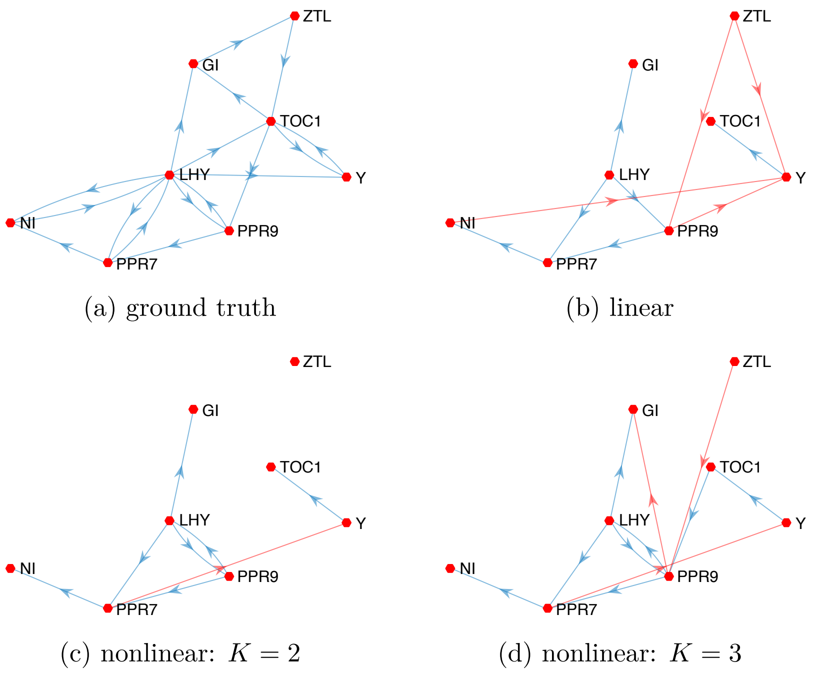
Figure 6.3. This simulation uses the Millar-10 model to simulate the measurements of key mRNAs for eukaryotic circadian clocks in experiments. This is "high" resolution data in experiments, whose sampling period is 30 min. Refer to the supplement of Pokhilko et al. (2010) for the nonlinear ordinary different equations.

The proposed method is used to perform network reconstruction. The whole time series is divided equidistantly into 1 (i.e. applying linear approach), 2, 3, 4 segments, i.e $K = 1, 2, 3, 4$. The results of $K = 1, 2, 3$ is shown in Figure 6.4. To determine whether an edge exists or not, we check the 2-norm of all parameters in its associated transfer function. Note that our approach has enhanced sparsity, and thus it is no longer an issue as other methods to select the threshold of 2-norm[3] (10^{-8} used in Figure 6.4) to determine whether an edge should exist or not, e.g. most methods reviewed in Aderhold et al. (2014).

As a convention on comparing different algorithms, the performance curves of the algorithm are given in Figure 6.5, and Figure 6.6 with multiple model orders. One can see that $K = 2$ in Figure 6.4 and Figure 6.5 clearly outperforms the others, which implies the choice of segmentation is important. There is a trade-off between the number of segmentation and the richness of each segment of time series. Increasing the number of segments in computation does better in approximating the ground truth Boolean dynamic network. However, this operation decreases the number of each segment of time series, which may then lead to bad performance of system identification. Unfortunately, we have not yet gained enough experience on how to perform segmentation of time series. Another point deserving to be stressed is that it may not be fair to compare sparse methods with other network reconstruction methods, which provide non-sparse score matrices and depend on careful selections of thresholds. We notice that, if decreasing the regularization parameter, the sparsity drops while the performance shown by ROC curves improves. Such an improvement is compromised in practice. Moreover, the performance of sparse methods shown by ROC fails to be impressive, due to the observation that the selection of thresholds concentrates on values smaller than 10^{-12}, which is fairly irregular and may fail the function of ROC curves. In applications, the author prefers to show performance by Figure 6.4 with a threshold of zero with respect to numerical precision (as the strategy used in Chapter 3). Choosing an "optimal" threshold is not practical and

[3]Any thresholds ranging from 10^{-6} to 10^{-11} gives the same results in Figure 6.4.

algorithms should ease this process. Admittedly, there is no agreement on which way is better to evaluate the performance.



(a) ground truth (b) linear

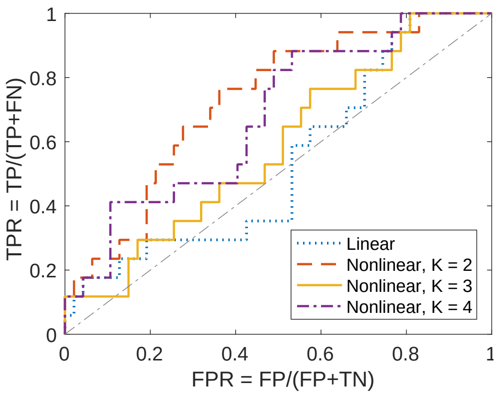
(c) nonlinear: $K = 2$ (d) nonlinear: $K = 3$

Figure 6.4. Examples of network inference results.

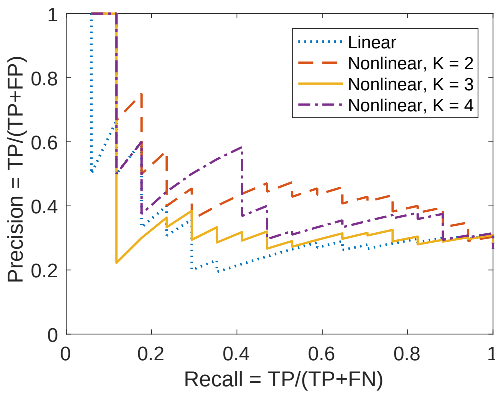
Considering the results in Figure 6.4, one may have noticed that the $K = 3$ case shows more wrong links than $K = 2$. This might probably be due to the deficiency of data. Two wrong links (PPR9 $\to$ GI, ZTL $\to$ PPR9) only come from the middle segment (-16h$\sim$16h), half of which is the responses of the plants under regular dark-light cycling conditions and the other half is under the irregularly constant lighting. However, neither of them has enough data points to deliver rich information. Nevertheless, this still points out a problem in this proposed method: high requirements on data (in the sense of biological data). Even though it does improve the performance, it cannot perform as good as we expected in theory due to the lack of data. Another problem deserving to be explained is why there are still several links missed. These links are introduced by the nonlinear terms combining x and u, i.e. nonlinear terms as $f(..., x_k, ...; u)$, which has not been included in our extension. The essential issue is that we have not known how to include bilinear terms of x and u into the DSF.

6.6 Conclusions

This chapter provides a way to define Boolean dynamic networks for a large relevant class of nonlinear dynamical systems. Based on this definition, an inference method was introduced with great practical applicability. To handle large-scale problems, an ADMM algorithm is presented. To show the performance of the network inference for nonlinear systems, an

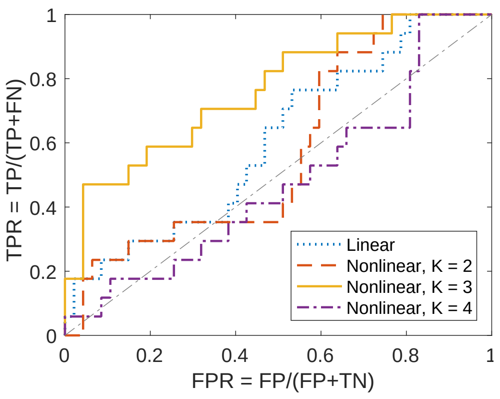


(a) ROC curve

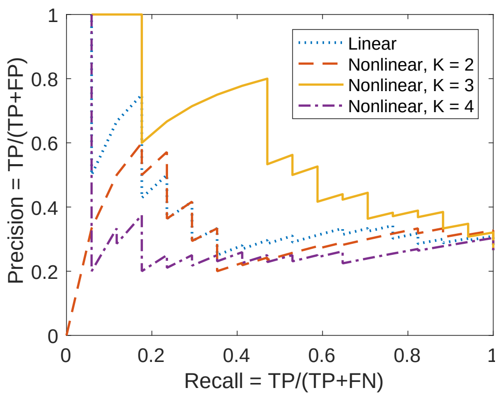


(b) P-R curve

Figure 6.5. Performance curves: the ROC and the precision-recall (P-R) curves. The score matrices in the computations are obtained by taking the 2-norm of the vector of parameters associated with each edge. For $K > 1$, the K number of network models are estimated and the score matrix is the average of score matrices of all models.



(a) ROC curve



(b) P-R curve

Figure 6.6. Averaged performance of different methods with model orders ranging from 1 to 48. This is not a plot of average of all ROC or P-R curves. It is obtained by taking average of score matrices of each model order before calculating performance indices.

example from the field of systems biology is used, the Millar-10 model. With this example we show how the new definition for nonlinear systems improves the inference of Boolean networks compared to the case when the linear approach is directly applied. Besides the definition and inference of Boolean dynamic networks for nonlinear systems, we also provide a way to define linear network identifiability, which uniforms and puts into context the available work on this topic.

Chapter 7

Conclusions

7.1 Summary

The dissertation starts with reviewing concepts on dynamical structure functions (DSF) and defining dynamic networks. In Chapter 2, we suggest a way to define linear network identifiability, which helps to unify the available work on network identifiability. The available results on network identifiability conditions are reviewed and commented. Further understanding and interpretations of DSF's are presented, such as hidden states, paths of DSF models.

The work in this dissertation focuses on methods and algorithms on dynamic network reconstruction. It is challenging due to the requirements on sparse network structures and the satisfaction of network identifiability. Moreover, the work stands on the practical demands of biological applications. The deficiency of data quality is taken into account: limited lengths of time series, heterogeneity of "replica" data, low sampling frequencies, nonlinearity, etc. These practical issues are the main driving force to reconstruction problems considered in this dissertation.

Heterogeneity of datasets in experiment repetition is inevitable due to variance of individuals and experiments subjected to different experimental conditions, e.g., changes/perturbations in parameters, disturbance or noise. We dedicate to exploring the common and essential properties, which are described as network structures. Chapter 3 addresses heterogeneous reconstruction problems of linear dynamic networks. The methods integrate the whole datasets and promote group sparsity to assure both network sparsity and the consistency of Boolean structures over datasets. Treatments by the iterative reweighted l_1 method and sparse Bayesian learning are provided, together with implementations via proximal methods and ADMM for large-dimensional networks.

The low sampling frequency makes identification problems particularly difficult. It forces us to estimate continuous-time models to reconstruct networks. However, due to limited experiment conditions, especially in biology, we have to work on low-sampling-frequency data. Chapter 4 raises the concept of "system aliasing" which plays a central role on the low-sampling-frequency reconstruction. A sampling theorem on no system aliasing is presented, together with a test criterion of system aliasing. In terms of no system aliasing cases, construction algorithms are provided for measurements with large sampling periods. Moreover, certain theoretical results are presented for the cases when system aliases are presented.

Due to particular difficulties on the calculation of gradients in parameter estimation for low-sampling-frequency cases, Chapter 5 uses the Expectation Minimization (EM) algorithm to bypass this issue, in which the M-step seems solving full-state measurement cases. Kalman filters and smoothers are adapted to estimate states. Parameters in each iteration are estimated by the maximum a posteriori (MAP), where the prior is constructed using Sparse Bayesian Learning (SBL) to enhance network sparsity. To solve the SBL problem, another EM algorithm is embedded, in which the constraints raised from network identifiability are integrated. We manage to offer a solution using Bayesian approaches to reconstruct dynamic networks from low-sampling-frequency data.

Systems in nature are inherently nonlinear. Chapter 6 presents a useful definition of Boolean dynamic networks for a large class of nonlinear systems. Moreover, a robust but approximated inference method is provided. The well-known Millar-10 model in systems biology is used as a numerical example, which provides the ground truth of causal networks for key mRNAs involved in eukaryotic circadian clocks.

In a sum, this dissertation provides a class of solutions to dynamic network reconstruction with considerations on practical issues in biological applications.

7.2 Outlook

This dissertation focuses on methods on dynamic network reconstruction with considerations on particular issues of data qualities in biological applications. However, many questions remain unsolved besides our study.

First, we comment on the general treatment to identification of LTI dynamic network models, i.e. DSF's, assuming the sampling frequency can be arbitrary high. This is not an easy problem, due to the difficulty on imposing sparsity. In a point of empirical view, determining network topology is particularly important. If time series of arbitrary lengths are available, it turns to be fairly easy due to the consistency properties of statistical estimation of DSF models, i.e. the zero elements in the ground truth of Q, P, H will converge to zero with the increase of lengths of time series. However, this is never the case, especially in biological applications. We have to resort to sparsity estimation to identify network structures from data of limited lengths. Unfortunately, we are restricted particular classes of time series models due to issues in numerical optimization. As shown in Chapter 3, algorithms are presented only for ARX parametrization of DSF's. We can easily parametrize DSF's by more general models, such as ARMAX, ARARX, ARARMAX, etc. (as ARMAX given in Chapter 3). However, we fail to provide reliable algorithms on sparse parameter estimation. The sparse estimation of nonlinear least square is still in process: SBL fails to work; solving l_1-regularized nonlinear least square remains challenging in numerical optimization and may fail to provide sparse results. Due to the restriction on choosing time series models (only ARX is available), as shown in numerical examples in Chapter 3, around 20% links in average cannot be constructed and it may get worse if decreasing SNR and data lengths.

Low sampling frequencies are another issue in practice that make reconstruction problem particularly challenging. In many biological applications, there are strict restrictions on increasing sampling frequencies, due to experiment conditions, sources, expenses, etc. It makes many classical methods fail to be plausible. Therefore, we have to revisit the identification/reconstruction problems as new ones in theory. However, the inclusion of matrix exponential and logarithm brings complexities on both theories and numerical computations. As shown in Chapter 4, it is not easy to solve even for the full-state measurement case, due to non-convexity and non-differentiality of resultant optimization problems. We are lack of methods in numerical optimization to solve non-convex and non-differential problems in general. Even if we provide algorithms for our particular problems, the performance is compromised and deserves more attentions. Chapter 5 uses EM and SBL methods. However, one may notice that we have to use matrix logarithm, which is an compromised choice due to the restriction of SBL. On the other hand, a simplified version (removing the logarithm part) of the method in Chapter 5 provides a state-space based method to identify (discrete-time) DSF models, which allow more general time series models than ARX parametrization. As the last topic studied in the project, it deserves more careful studies.

Nonlinearity is another inevitable issue in dynamic network reconstruction. One widely used definition is Bayesian networks defined based on conditional independency, which in theory can handle nonlinearity but require assumptions on Markovian property. As an alternative, in a point of system theoretic view, Chapter 6 shows a tentative definition and presents an approximated reconstruction methods. A lot of questions on this definition tend to be solved. For instance, what dynamic systems in Boolean network reconstruction can be resolved by choosing one operating point and how to perturb the systems to collect rich enough data; what dynamic systems can be resolved by choosing a finite number of operating points. Moreover, the class of nonlinear state-space models used in Chapter 6 has not yet been general enough. How to extend this definition to boarder classes remains unsolved, particularly including bilinear terms.

Appendix A

Matrix Functions

A.1 Matrix Exponential and Logarithm

Theorem A.1 (Gantmacher (Higham, 2008a, Thm. 1.27)). *Let $P \in \mathbb{C}^{n\times n}$ be nonsingular with the Jordan canonical form*

$$Z^{-1}PZ = J = \operatorname{diag}(J_1, J_2, ..., J_p) \tag{A.1a}$$

$$J_k = J_k(\lambda_k) = \begin{bmatrix} \lambda_k & 1 & & \\ & \lambda_k & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k \end{bmatrix} \in \mathbb{C}^{m_k\times m_k}. \tag{A.1b}$$

Then all solutions to $e^A = P$ are given by

$$A = ZU \operatorname{diag}(L_1^{j_1}, L_2^{j_2}, ..., L_p^{j_p})U^{-1}Z^{-1}, \tag{A.2}$$

where

$$L_k^{j_k} = \log(J_k(\lambda_k)) + 2j_k\pi i I_{m_k}; \tag{A.3}$$

$\log(J_k(\lambda_k))$ *denotes*

$$f(J_k) := \begin{bmatrix} f(\lambda_k) & f'(\lambda_k) & \cdots & \frac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{bmatrix}$$

with f the principal branch of the logarithm, defined by $\operatorname{Im}(\log(z)) \in (-\pi, \pi]$; j_k is an arbitrary integer; and U is an arbitrary nonsingular matrix that commutes with J.

Theorem A.2 (classification of logarithms (Higham, 2008a, Thm. 1.28)). *Let the nonsingular matrix $P \in \mathbb{C}^{n\times n}$ have the Jordan canonical form (A.1) with p Jordan blocks, and let $s \le p$ be the number of distinct eigenvalues of A. Then $e^A = P$ has a countable infinity of solutions that are primary functions of P, given by*

$$A_j = Z\,\mathrm{diag}(L_1^{j_1}, L_2^{(j_2)}, ..., L_p^{(j_p)})Z^{-1}, \tag{A.4}$$

where $L_k^{j_k}$ is defined in (A.3), corresponding to all possible choices of the integers $j_1, ..., j_p$, subject to the constraint that $j_i = j_k$ whenever $\lambda_i = \lambda_k$.

If $s < p$ then $e^A = P$ has nonprimary solutions. They form parametrized families

$$A_j(U) = ZU\,\mathrm{diag}(L_1^{j_1}, L_2^{(j_2)}, ..., L_p^{(j_p)})U^{-1}Z^{-1}, \tag{A.5}$$

where j_k is an arbitrary integer, U is an arbitrary nonsingular matrix that commutes with J, and for each j there exist i and k, depending on j, such that $\lambda_i = \lambda_k$ while $j_i \neq j_k$.

A.2 Fréchet Derivatives

Definition A.3 (Fréchet Derivatives (Higham, 2008a)). The Fréchet derivative of the matrix function $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ at a point $X \in \mathbb{C}^{(n\times n)}$ is a linear mapping

$$\mathbb{C}^{n\times n} \xrightarrow{L} \mathbb{C}^{n\times n}$$
$$E \longmapsto L(A, E)$$

such that for all $E \in \mathbb{C}^{n\times n}$

$$f(A + E) - f(A) - L(A, E) = o(\|E\|).$$

The Fréchet derivative is unique if it exists, and for matrix functions exp (matrix exponential) and Log (principal matrix logarithm) it exists. The Fréchet derivative of the function exp is

$$L_{\exp}(X, E) = \int_0^1 e^{X(1-s)} E e^{Xs} ds, \tag{A.6}$$

which can be efficiently calculated by the *Scaling-Pade-Squaring* method in Al-Mohy and Higham (2009). It gives a linear approximation of exp at a given point A_c in the direction E

$$e^{hA} = e^{h(A_c+E)} = e^{hA_c} + L(hA_c, hE) + O(\|hE\|^2). \tag{A.7}$$

The Fréchet derivative of the function Log is

$$L_{\mathrm{Log}}(X,E) = \int_0^1 (t(X-I)+I)^{-1}E(t(X-I)+I)^{-1})dt,$$

and its efficient computation algorithm is provided in Al-Mohy et al. (2013).

Theorem A.4 (Kronecker representation (Higham, 2008a, Thm. 10.13)). *For* $A \in \mathbb{C}^{n\times n}$, $\mathrm{vec}(L(A,E)) = K(A)\,\mathrm{vec}(E)$, *where* $K(A) \in \mathbb{C}^{n^2\times n^2}$ *has the representations*

$$K(A) = \begin{cases} (I \otimes e^{A})\psi\left(A^T \oplus (-A)\right) \\ (e^{A^T/2} \otimes e^{A/2})\,\mathrm{sinch}\left(\frac{1}{2}[A^T \oplus (-A)]\right) \\ \frac{1}{2}(e^{A^T} \oplus e^{A})\tau\left(\frac{1}{2}[A^T \oplus (-A)]\right) \end{cases}$$

where $\psi(x) = (e^x - 1)/x$ *and* $\tau(x) = \tanh(x)/x$. *The third expression is valid if* $\frac{1}{2}\|A^T \oplus (-A)\| < \pi/2$ *for some consistent matrix norm.*

Here the operator $\oplus$ is the *Kronecker sum*, defined as

$$(A \oplus B) = A \otimes I_n + I_m \otimes B,$$

for $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{n\times n}$.

References

Aderhold, A., Husmeier, D., and Grzegorczyk, M. (2014). Statistical inference of regulatory networks for circadian regulation.

Äijö, T. and Lähdesmäki, H. (2009). Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. *Bioinformatics*, 25(22):2937–2944.

Akutsu, T., Miyano, S., and Kuhara, S. (1999). Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. In *Pacific symposium on biocomputing*, volume 4, pages 17–28. Citeseer.

Akutsu, T., Miyano, S., and Kuhara, S. (2000). Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics*, 16(8):727–734.

Al-Mohy, A. H. and Higham, N. J. (2009). Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimation. *SIAM Journal on Matrix Analysis and Applications*, 30(4):1639–1657.

Al-Mohy, A. H., Higham, N. J., and Relton, S. D. (2013). Computing the Fréchet derivative of the matrix logarithm and estimating the condition number. *SIAM Journal on Scientific Computing*, 35(4):C394–C410.

Alberts, B. (2015). *Molecular biology of the cell.* Garland Science, Taylor and Francis Group, New York, NY.

Amit, I., Garber, M., Chevrier, N., Leite, A. P., Donner, Y., Eisenhaure, T., Guttman, M., Grenier, J. K., Li, W., and Zuk, O. (2009). Unbiased reconstruction of a mammalian transcriptional network mediating pathogen responses. *Science*, 326(5950):257–263.

Arbeitman, M. N., Furlong, E. E. M., Imam, F., Johnson, E., Null, B. H., Baker, B. S., Krasnow, M. A., Scott, M. P., Davis, R. W., and White, K. P. (2002). Gene expression during the life cycle of Drosophila melanogaster. *Science*, 297(5590):2270–2275.

Åström, K. J. (1980). Maximum likelihood and prediction error methods. *Automatica*, 16(5):551–574.

Åström, K. J. (2012). *Introduction to stochastic control theory.* Courier Corporation.

Bar-Joseph, Z., Gitter, A., and Simon, I. (2012). Studying and modelling dynamic biological processes using time-series gene expression data. *Nature Reviews Genetics*, 13(8):552–564.

Bar-Joseph, Z., Siegfried, Z., Brandeis, M., Brors, B., Lu, Y., Eils, R., Dynlacht, B. D., and Simon, I. (2008). Genome-wide transcriptional analysis of the human cell cycle identifies genes differentially regulated in normal and cancer cells. *Proceedings of the National Academy of Sciences*, 105(3):955–960.

Baraniuk, R. G. (2007). Compressive sensing. *IEEE signal processing magazine*, 24(4).

Beal, M. J., Falciani, F., Ghahramani, Z., Rangel, C., and Wild, D. L. (2005). A Bayesian approach to reconstructing genetic regulatory networks with hidden factors. *Bioinformatics*, 21(3):349–356.

Berger, J. O. (1993). *Statistical Decision Theory and Bayesian Analysis (Springer Series in Statistics).* Springer.

Bi, J., Bennett, K., Embrechts, M., Breneman, C., and Song, M. (2003). Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3(Mar):1229–1243.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization.* Cambridge university press.

Brockwell, P. J. and Davis, R. A. (2009). *Time series: theory and methods.* Springer Science & Business Media.

Buntine, W. (1991). Theory refinement on Bayesian networks. In *Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann Publishers Inc.

Buzsaki, G. (2011). *Rhythms of the Brain.* Oxford University Press.

Candes, E. J. and Plan, Y. (2011). A probabilistic and RIPless theory of compressed sensing. *IEEE Transactions on Information Theory*, 57(11):7235–7254.

Candes, E. J. and Tao, T. (2005). Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215.

Candes, E. J., Wakin, M. B., and Boyd, S. P. (2008). Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905.

Chambolle, A. (2004). An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1):89–97.

Chang, C. and Glover, G. H. (2010). Time–frequency dynamics of resting-state brain connectivity measured with fMRI. *Neuroimage*, 50(1):81–98.

Chartrand, R. and Yin, W. (2008). Iteratively reweighted algorithms for compressive sensing. In *Acoustics, speech and signal processing, 2008. ICASSP 2008. IEEE international conference on*, pages 3869–3872. IEEE.

Chen, T., Ohlsson, H., and Ljung, L. (2012). On the estimation of transfer functions, regularizations and Gaussian processes—revisited. *Automatica*, 48(8):1525–1535.

Chetty, V. and Warnick, S. (2015). Network semantics of dynamical systems. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 1557–1562. IEEE.

Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90(432):1313–1321.

Chickering, D. M. (1996). Learning Bayesian networks is NP-complete. *Learning from data: Artificial intelligence and statistics V*, 112:121–130.

Chiuso, A. and Pillonetto, G. (2012). A Bayesian approach to sparse dynamic network identification. *Automatica*, 48(8):1553–1565.

Combettes, P. L. and Pesquet, J.-C. (2011). Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer.

Cooper, G. F. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347.

Csurcsia, P. Z. and Lataire, J. (2016). Nonparametric Estimation of Time-Varying Systems Using 2-D Regularization. *IEEE Transactions on Instrumentation and Measurement*, 65(5):1259–1270.

CVX Research, I. (2008). CVX: Matlab software for disciplined convex programming.

Dahlhaus, R. and Eichler, M. (2003). Causality and graphical models in time series analysis. *Oxford Statistical Science Series*, pages 115–137.

Dean, T. L. and Kanazawa, K. (1988). Probabilistic Temporal Reasoning. In *AAAI*, pages 524–529.

Dinuzzo, F. (2015). Kernels for linear time invariant system identification. *SIAM Journal on Control and Optimization*, 53(5):3299–3317.

Donoho, D. L. (2006). Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306.

Dougherty, E. R., Kim, S., and Chen, Y. (2000). Coefficient of determination in nonlinear signal processing. *Signal Processing*, 80(10):2219–2235.

Eguiluz, V. M., Chialvo, D. R., Cecchi, G. A., Baliki, M., and Apkarian, A. V. (2005). Scale-free brain functional networks. *Physical review letters*, 94(1):18102.

Eichler, M. (2000). Granger-causality graphs for multivariate time series. *Preprint, University of Heidelberg*.

Eichler, M. (2006). On the evaluation of information flow in multivariate systems by the directed transfer function. *Biological Cybernetics*, 94:469–482.

Eichler, M. (2007). Granger causality and path diagrams for multivariate time series. *Journal of Econometrics*, 137:334–353.

Feist, A. M., Herrgård, M. J., Thiele, I., Reed, J. L., and Palsson, B. Ø. (2009). Reconstruction of biochemical networks in microorganisms. *Nature Reviews Microbiology*, 7(2):129–143.

Fogelmark, K. and Troein, C. (2014). Rethinking transcriptional activation in the Arabidopsis circadian clock. *PLoS computational biology*, 10(7):e1003705.

Fornito, A., Zalesky, A., and Bullmore, E. (2016). *Fundamentals of Brain Network Analysis*. Academic Press.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*.

Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using Bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620.

Friedman, N., Murphy, K., and Russell, S. (1998). Learning the structure of dynamic probabilistic networks. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 139–147. Morgan Kaufmann Publishers Inc.

Friston, K. J., Harrison, L., and Penny, W. (2003). Dynamic causal modelling. *Neuroimage*, 19(4):1273–1302.

Gall, J.-F. L. (2016). *Brownian Motion, Martingales, and Stochastic Calculus (Graduate Texts in Mathematics)*. Springer, 1st ed. 20 edition.

Gardner, T. S., Di Bernardo, D., Lorenz, D., and Collins, J. J. (2003). Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105.

Garnier, H., Mensler, M., and Richard, A. (2003). Continuous-time model identification from sampled data: implementation issues and performance evaluation. *International Journal of Control*, 76(13):1337–1357.

Garnier, H. and Wang, L. (2008). *Identification of Continuous-time Models from Sampled Data*. Springer London.

Gasch, A. P., Spellman, P. T., Kao, C. M., Carmel-Harel, O., Eisen, M. B., Storz, G., Botstein, D., and Brown, P. O. (2000). Genomic expression programs in the response of yeast cells to environmental changes. *Molecular biology of the cell*, 11(12):4241–4257.

Gevers, M., Bazanella, A. S., and Parraga, A. (2016). Structural conditions for the identifiability of dynamical networks. In *IEEE Conference on Decision and Control (preprint)*.

Geweke, J. (1978). Testing the exogeneity specification in the complete dynamic simultaneous equation model. *Journal of Econometrics*, 7(2):163–185.

Gibson, S. and Ninness, B. (2005). Robust maximum-likelihood estimation of multivariable dynamic systems. *Automatica*, 41(10):1667–1682.

Glass, L. and Kauffman, S. A. (1973). The logical analysis of continuous, non-linear biochemical control networks. *Journal of theoretical Biology*, 39(1):103–129.

Goncalves, J. and Warnick, S. (2008). Necessary and Sufficient Conditions for Dynamical Structure Reconstruction of LTI Networks. *Automatic Control, IEEE Transactions on*, 53(7):1670–1674.

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438.

Grant, M. (2004). *Disciplined Convex Programming.* Phd thesis, Stanford University.

Grant, M. C. and Boyd, S. P. (2015). *The CVX Users' Guide, Release v2.1.*

Greenham, K. and McClung, C. R. (2015). Integrating circadian dynamics with physiological processes in plants. *Nature Reviews Genetics*, 16(10):598–610.

Hagmann, P., Cammoun, L., Gigandet, X., Meuli, R., Honey, C. J., Wedeen, V. J., and Sporns, O. (2008). Mapping the structural core of human cerebral cortex. *PLoS biology*, 6(7):e159.

Hannan, E. J. (1969). The identification of vector mixed autoregressive-moving average system. *Biometrika*, 56(1):223–225.

Hayden, D., Chang, Y. H., Goncalves, J., and Tomlin, C. J. (2016a). Sparse network identifiability via Compressed Sensing. *Automatica*, 68:9–17.

Hayden, D., Yuan, Y., and Goncalves, J. (2016b). Network Identifiability from Intrinsic Noise. *IEEE Transactions on Automatic Control*, PP(99):1.

He, F., Chen, H., Probst-Kepper, M., Geffers, R., Eifes, S., del Sol, A., Schughart, K., Zeng, A., and Balling, R. (2012). PLAU inferred from a correlation network is critical for suppressor function of regulatory T cells. *Molecular Systems Biology*, 8(1).

Heckerman, D. (1998). A tutorial on learning with Bayesian networks. In *Learning in graphical models*, pages 301–354. Springer.

Heckerman, D. and Geiger, D. (1995). Likelihoods and parameter priors for Bayesian networks. *Tech. MSRTR-95-54. Microsoft Research.*

Heckerman, D., Mamdani, A., and Wellman, M. P. (1995). Real-world applications of Bayesian networks. *Communications of the ACM*, 38(3):24–26.

Higham, N. (2008a). *Functions of Matrices.* Society for Industrial and Applied Mathematics.

Higham, N. J. (2008b). The Matrix Function Toolbox: A MATLAB toolbox connected with functions of matrices.

Honey, C. J., Sporns, O., Cammoun, L., Gigandet, X., Thiran, J.-P., Meuli, R., and Hagmann, P. (2009). Predicting human resting-state functional connectivity from structural connectivity. *Proceedings of the National Academy of Sciences*, 106(6):2035–2040.

Horn, R. A. and Piepmeyer, G. G. (2003). Two applications of the theory of primary matrix functions. *Linear Algebra and its Applications*, 361:99–106.

Hsiao, C. (1982). Autoregressive modeling and causal ordering of economic variables. *Journal of Economic Dynamics and Control*, 4:243–259.

Karatzas, I. and Shreve, S. (2012). *Brownian motion and stochastic calculus*, volume 113. Springer Science & Business Media.

Karlebach, G. and Shamir, R. (2008). Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9(10):770–780.

Kass, R., Tierney, L., and Kadane, J. (1988). Asymptotics in Bayesian computation. *Bayesian statistics*, 3:261–278.

Kauffman, S. A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, 22(3):437–467.

Kholodenko, B. N., Kiyatkin, A., Bruggeman, F. J., Sontag, E., Westerhoff, H. V., and Hoek, J. B. (2002). Untangling the wires: a strategy to trace functional interactions in signaling and gene networks. *Proceedings of the National Academy of Sciences*, 99(20):12841–12846.

Kim, S., Dougherty, E. R., Chen, Y., Sivakumar, K., Meltzer, P., Trent, J. M., and Bittner, M. (2000). Multivariate measurement of gene expression relationships. *Genomics*, 67(2):201–209.

Kuo, L. and Mallick, B. (1998). Variable selection for regression models. *Sankhya: The Indian Journal of Statistics, Series B*, pages 65–81.

Lee, J. M. (2012). *Introduction to Smooth Manifolds.* Graduate Texts in Mathematics 218. Springer-Verlag New York, 2 edition.

Liu, J. S. (2008). *Monte Carlo strategies in scientific computing.* Springer Science & Business Media.

Ljung, L. (1999). *System Identification: Theory for the User.* Prentice-Hall information and system sciences series. Prentice Hall PTR.

Ljung, L. and Wills, A. (2010). Issues in sampling and estimating continuous-time models with stochastic disturbances. *Automatica*, 46(5):925–931.

Locke, J. C. W., Kozma-Bognár, L., Gould, P. D., Fehér, B., Kevei, E., Nagy, F., Turner, M. S., Hall, A., and Millar, A. J. (2006). Experimental validation of a predicted feedback loop in the multi-oscillator clock of Arabidopsis thaliana. *Molecular systems biology*, 2(1):59.

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural computation*, 4(3):415–447.

Marbach, D., Costello, J. C., Küffner, R., Vega, N. M., Prill, R. J., Camacho, D. M., Allison, K. R., Kellis, M., Collins, J. J., and Stolovitzky, G. (2012). Wisdom of crowds for robust gene network inference. *Nature methods*, 9(8):796–804.

Marbach, D., Prill, R. J., Schaffter, T., Mattiussi, C., Floreano, D., and Stolovitzky, G. (2010). Revealing strengths and weaknesses of methods for gene network inference. *Proceedings of the National Academy of Sciences of the United States of America*, 107(14):6286–6291.

Margolin, A. A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Dalla Favera, R., and Califano, A. (2006). ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC bioinformatics*, 7 Suppl 1:S7.

Masani, P. (1966). Recent trends in multivariate prediction theory. In Krishnaiah, P., editor, *Multivariate Analysis I*, pages 351–382. Academic press New York.

Mason, S. J. (1953). *Feedback Theory: I. Some Properties of Signal Flow Graphs.* Massachusetts Institute of Technology, Research Laboratory of Electronics.

Mason, S. J. (1956). *Feedback theory: further properties of signal flow graphs.* Research Laboratory of Electronics, Massachusetts Institute of Technology.

Materassi, D. and Innocenti, G. (2010). Topological identification in networks of dynamical systems. *Automatic Control, IEEE Transactions on*, 55(8):1860–1871.

Materassi, D. and Salapaka, M. V. (2012). Network reconstruction of dynamical polytrees with unobserved nodes. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 4629–4634. IEEE.

MathWorks (2016). Regularized Identification of Dynamic Systems.

McClung, C. R. (2006). Plant circadian rhythms. *The Plant Cell*, 18(4):792–803.

Moreno, E. L., Hachi, S., Hemmer, K., Trietsch, S. J., Baumuratov, A. S., Hankemeier, T., Vulto, P., Schwamborn, J. C., and Fleming, R. M. T. (2015). Differentiation of neuroepithelial stem cells into functional dopaminergic neurons in 3D microfluidic cell culture. *Lab on a Chip*, 15(11):2419–2428.

Mörters, P. and Peres, Y. (2010). *Brownian Motion.* Cambridge University Press, 1 edition.

Murphy, K. and Mian, S. (1999). Modelling gene expression data using dynamic Bayesian networks. Technical report.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective.* MIT press.

Nabavi, S., Chakrabortty, A., and Khargonekar, P. P. (2015). A global identifiability condition for consensus networks on tree graphs.

Palsson, B. Ø. (2011). *Systems biology: simulation of dynamic network states.* Cambridge University Press.

Pan, W., Yuan, Y., Goncalves, J., and Stan, G.-B. (2016). A Sparse Bayesian Approach to the Identification of Nonlinear State-Space Systems. *Automatic Control, IEEE Transactions on*, 61(1):182–187.

Pan, W., Yuan, Y., Ljung, L., Gon, J., and Stan, G.-B. (2015). Identifying biochemical reaction networks from heterogeneous datasets. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 2525–2530. IEEE.

Parikh, N. and Boyd, S. (2013). Proximal algorithms. *Foundations and Trends in optimization*, 1(3):123–231.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (Morgan Kaufmann Series in Representation and Reasoning).* Morgan Kaufmann.

Pearl, J. (2000). *Causality: models, reasoning and inference*, volume 29. Cambridge Univ Press.

Pillonetto, G., Chiuso, A., and De Nicolao, G. (2011). Prediction error identification of linear systems: a nonparametric Gaussian regression approach. *Automatica*, 47(2):291–305.

Pillonetto, G. and De Nicolao, G. (2010). A new kernel-based approach for linear system identification. *Automatica*, 46(1):81–93.

Pillonetto, G., Dinuzzo, F., Chen, T., De Nicolao, G., and Ljung, L. (2014). Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*, 50(3):657–682.

Pokhilko, A., Hodge, S. K., Stratford, K., Knox, K., Edwards, K. D., Thomson, A. W., Mizuno, T., and Millar, A. J. (2010). Data assimilation constrains new connections and components in a complex, eukaryotic circadian clock model. *Molecular systems biology*, 6(1):416.

Raftery, A. E. (1996). Hypothesis testing and model selection via posterior simulation. *Markov chain Monte Carlo in practice*, pages 163–188.

Reed, M. and Simon, B. (1972). *Methods of modern mathematical physics. vol. 1. Functional analysis*, volume 2. Academic press New York.

Renault, E. and Triacca, U. (2015). Causality and separability. *Statistics & Probability Letters.*

Richardson, T. (2003). Markov properties for acyclic directed mixed graphs. *Scandinavian Journal of Statistics*, 30(1):145–157.

Rogers, S. and Girolami, M. (2005). A Bayesian regression approach to the inference of regulatory networks from gene expression data. *Bioinformatics*, 21(14):3131–3137.

Rozanov, Y. A. (1967). *Stationary random processes.* Holden-Day.

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268.

Rustici, G., Mata, J., Kivinen, K., Lió, P., Penkett, C. J., Burns, G., Hayles, J., Brazma, A., Nurse, P., and Bähler, J. (2004). Periodic gene expression program of the fission yeast cell cycle. *Nature genetics*, 36(8).

Sahiner, B., Chen, W., Pezeshk, A., and Petrick, N. (2017). Comparison of two classifiers when the data sets are imbalanced: the power of the area under the precision-recall curve as the figure of merit versus the area under the ROC curve. In *SPIE Medical Imaging*, pages 101360G–101360G. International Society for Optics and Photonics.

Shmulevich, I., Dougherty, E. R., Kim, S., and Zhang, W. (2002). Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274.

Shumway, R. H. and Stoffer, D. S. (2017). *Time Series Analysis and Its Applications: With R Examples (Springer Texts in Statistics)*. Springer, 4th ed. 20 edition.

Simon, N. and Tibshirani, R. (2012). Standardization and the group lasso penalty. *Statistica Sinica*, 22(3):983.

Sims, C. A. (1972). Money, income, and causality. *The American economic review*, pages 540–552.

Skoog, G. R. (1976). Causality Characterizations: Bivariate, Trivariate, and Multivariate Propositions. Technical report.

Skudlarski, P., Jagannathan, K., Calhoun, V. D., Hampson, M., Skudlarska, B. A., and Pearlson, G. (2008). Measuring brain connectivity: diffusion tensor imaging validates resting state temporal correlations. *Neuroimage*, 43(3):554–561.

Song, M. J., Lewis, C. K., Lance, E. R., Chesler, E. J., Yordanova, R. K., Langston, M. A., Lodowski, K. H., and Bergeson, S. E. (2009). Reconstructing generalized logical networks of transcriptional regulation in mouse brain from temporal gene expression data. *EURASIP J.Bioinform.Syst.Biol*, 2009:545176.

Sontag, E. D. (2008). Network reconstruction based on steady-state data. *Essays in Biochemistry*, 45:161–176.

Stam, C. J. (2004). Functional connectivity patterns of human magnetoencephalographic recordings: a 'small-world'network? *Neuroscience letters*, 355(1):25–28.

Tegner, J., Yeung, M. K. S., Hasty, J., and Collins, J. J. (2003). Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling. *Proceedings of the National Academy of Sciences*, 100(10):5944–5949.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society.Series B (Methodological)*, pages 267–288.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *The journal of machine learning research*, 1:211–244.

Van den Hof, P. M. J., Dankers, A., Heuberger, P. S. C., and Bombois, X. (2013). Identification of dynamic models in complex networks with prediction error methods-Basic methods for consistent module estimates. *Automatica*, 49(10):2994–3006.

Van den Hof, P. M. J., Dankers, A. G., and Weerts, H. H. M. (2017). From closed-loop identification to dynamic networks: generalization of the direct method.

Van Overschee, P. and De Moor, B. L. (2012). *Subspace identification for linear systems: Theory—Implementation—Applications*. Springer Science & Business Media.

van Someren, E. P., Wessels, L. F. A., and Reinders, M. J. T. (2000). Linear modeling of genetic networks from experimental data. In *Ismb*, pages 355–366.

Viberg, M. (2002). Subspace-based state-space system identification. *Circuits, Systems and Signal Processing*, 21(1):23–37.

Warnick, S. (2015). Shared Hidden State and Network Representations of Interconnected Dynamical Systems. In *53rd Annual Allerton Conference on Communications, Control, and Computing*, Monticello, IL.

Weerts, H., Van den Hof, P. M. J., and Dankers, A. (2016). Identification of dynamic networks operating in the presence of algebraic loops. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 4606–4611. IEEE.

Weerts, H., Van den Hof, P. M. J., and Dankers, A. (2018a). Single module identifiability in linear dynamic networks. *arXiv preprint arXiv:1803.02586*.

Weerts, H. H. M., Dankers, A. G., and Van den Hof, P. M. J. (2015). Identifiability in dynamic network identification. *IFAC-PapersOnLine*, 48(28):1409–1414.

Weerts, H. H. M., Van den Hof, P. M. J., and Dankers, A. G. (2018b). Identifiability of linear dynamic networks. *Automatica*, 89:247–258.

Willems, J. C. (2007). The behavioral approach to open and interconnected systems. *Control Systems, IEEE*, 27(6):46–99.

Wipf, D. and Nagarajan, S. (2010). Iterative reweighted l1 and l2 methods for finding sparse solutions. *Selected Topics in Signal Processing, IEEE Journal of*, 4(2):317–329.

Wipf, D. P. and Rao, B. D. (2004). Sparse Bayesian learning for basis selection. *Signal Processing, IEEE Transactions on*, 52(8):2153–2164.

Wipf, D. P. and Rao, B. D. (2007). An empirical Bayesian strategy for solving the simultaneous sparse approximation problem. *Signal Processing, IEEE Transactions on*, 55(7):3704–3716.

Wipf, D. P., Rao, B. D., and Nagarajan, S. (2011). Latent variable Bayesian models for promoting sparsity. *Information Theory, IEEE Transactions on*, 57(9):6236–6255.

Woodbury, N., Dankers, A., and Warnick, S. (2017). On the well-posedness of LTI networks. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 4813–4818. IEEE.

Yeung, K. Y., Bumgarner, R. E., and Raftery, A. E. (2005). Bayesian model averaging: development of an improved multi-class, gene selection and classification tool for microarray data. *Bioinformatics (Oxford, England)*, 21(10):2394–2402.

Yu, J., Smith, V. A., Wang, P. P., Hartemink, A. J., and Jarvis, E. D. (2004). Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–3603.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Yuan, Y., Glover, K., and Gonçalves, J. (2015). On minimal realisations of dynamical structure functions. *Automatica*, 55:159–164.

Yue, Z. (2018). *Dynamic Network Reconstruction in Systems Biology: Methods and Algorithms*. Ph.d. dissertation, University of Luxembourg.

Yue, Z., Pan, W., Thunberg, J., Ljung, L., and Goncalves, J. (2017). Linear Dynamic Network Reconstruction from Heterogeneous Datasets. In *Preprints of the 20th World Congress, IFAC*, pages 11075–11080, Toulouse, France.

Yue, Z., Thunberg, J., and Goncalves, J. (2016a). Inverse Problems for Matrix Exponential in System Identification: System Aliasing. In *2016 22nd Proceedings of International Symposium on Machematical Theory of Networks and Systems.*

Yue, Z., Thunberg, J., Ljung, L., Yuan, Y., and Goncalves, J. (2016b). Systems Aliasing in Dynamic Network Reconstruction: Issues on Low Sampling Frequencies. *Ready for submission (arXiv 1605.08590).*

Yue, Z., Thunberg, J., Pan, W., Ljung, L., and Goncalves, J. (2018). Dynamic Network Reconstruction from Heterogeneous Datasets. *Ready for submission (arXiv 1612.01963).*

Yuh, C.-H., Bolouri, H., and Davidson, E. H. (1998). Genomic cis-regulatory logic: experimental and computational analysis of a sea urchin gene. *Science*, 279(5358):1896–1902.

Zalesky, A., Fornito, A., and Bullmore, E. (2012). On the use of correlation as a measure of network connectivity. *Neuroimage*, 60(4):2096–2106.

Zhou, K. and Doyle, J. C. (1998). *Essentials of robust control.* Prentice Hall, Upper Saddle River, N.J.

Zhou, K., Doyle, J. C., and Glover, K. (1996). *Robust and optimal control*, volume 40. Prentice Hall New Jersey.

Zhu, J., Rosset, S., Tibshirani, R., and Hastie, T. J. (2004). 1-norm support vector machines. In *Advances in neural information processing systems*, pages 49–56.

Index