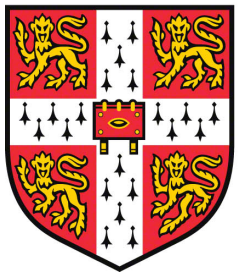
Novel methods for biological network inference: an application to circadian Ca^{2+} signaling network



University of Cambridge

Downing College

by

Junyang Jin

Supervisor:
Prof. Alex Webb

Co-Supervisor:
Prof. Jorge Gonçalves

May 2018

A dissertation submitted for the degree of

Doctor of Philosophy

Summary

Novel methods for biological network inference: an application to circadian Ca^{2+} signaling network

Junyang Jin

Biological processes involve complex biochemical interactions among a large number of species like cells, RNA, proteins and metabolites. Learning these interactions is essential to interfering artificially with biological processes in order to, for example, improve crop yield, develop new therapies, and predict new cell or organism behaviors to genetic or environmental perturbations. For a biological process, two pieces of information are of most interest. For a particular species, the first step is to learn which other species are regulating it. This reveals topology and causality. The second step involves learning the precise mechanisms of how this regulation occurs. This step reveals the dynamics of the system. Applying this process to all species leads to the complete dynamical network. Systems biology is making considerable efforts to learn biological networks at low experimental costs. The main goal of this thesis is to develop advanced methods to build models for biological networks, taking the circadian system of *Arabidopsis thaliana* as a case study. A variety of network inference approaches have been proposed in the literature to study dynamic biological networks. However, many successful methods either require prior knowledge of the system or focus more on topology. This thesis presents novel methods that identify both network topology and dynamics, and do not depend on prior knowledge. Hence, the proposed methods are applicable to general biological networks. These methods are initially developed for linear systems, and, at the cost of higher computational complexity, can also be applied to nonlinear systems. Overall, we propose four methods with increasing computational complexity: one-to-one, combined group and element sparse Bayesian learning (GESBL), the kernel method and reversible jump Markov chain Monte Carlo method (RJMCMC). All methods are tested with challenging dynamical network simulations (including feedback, random networks, different levels of noise and number of samples), and realistic models of circadian system of *Arabidopsis thaliana*. These simulations show that, while the one-to-one method scales to the whole genome, the kernel method and RJMCMC method are superior for smaller networks. They are robust to tuning variables and able to provide stable performance. The simulations also imply the advantage of GESBL and RJMCMC over the state-of-the-art method. We envision that the estimated models can benefit a wide range of research. For example, they can locate biological compounds responsible for human disease through mathematical analysis and help predict the effectiveness of new treatments.

Acknowledgement

I would like to express my sincere gratitude to my supervisors, Prof. Alex Webb and Prof. Jorge Gonçalves, for their patient guidance and professional instructions on biological and mathematical aspects. Without them, it would be impossible for me to combine these two fields of research. Their supervision has been crucial for me in accomplishing this project.

I would like to thank the circadian signal transduction group. The group members have offered generous help in setting up and guiding me through experiments. Their professional knowledge of biology also helped me a lot during my research.

I also spent much time visiting the systems control group of Luxembourg Centre for Systems Biomedicine. Many of the group members are dedicated in the theoretical development of network inference. I am very grateful to them for sharing their latest research outcomes. I have benefited a lot from the discussion with these researchers.

In addition, I want to especially thank Prof. Ye Yuan, the former PhD student of Prof. Jorge Gonçalves, who provided me his expertise in control engineering and kindly invited me to visit his lab in China.

Finally, I really appreciate my parents' and friends' support during my PhD life. Without them, I could hardly have made any progress.

Disclaimer

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text.

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 Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. 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 Cambridge or any other University or similar institution except as declared in the Preface and specified in the text

It does not exceed the prescribed word limit for the relevant Degree Committee.

Junyang Jin
Downing College
May 2018

Contents

Chapter 1.

Introduction

Biological networks form an essential part of the real world from ecosystems in nature, over cell systems in plants, to neural networks of human body. Although there are diverse biological networks, their operation involves some principle systems, including metabolic networks, signaling networks, protein-protein interaction networks, gene regulatory networks and gene co-expression networks. Identifying these networks is fundamental to interfering artificially with biological processes for the purpose of developing therapies to treat disease, improving crop yield and predicting cell or organism behaviors to genetic and environmental perturbations. For example, the discovery that neurodegeneration is a regulated cell-autonomous process of programmed cell death highly supports the development of protective therapies to Parkinson's disease [1], [2]. Another example is genetically-modified crops that are studied to meet the worldwide demand for quality foods [3], [4]. While learning these networks is important, few such networks are understood with complete structure and less is known on their dynamical behavior over time. Hence, it is necessary to seek effective and efficient ways to infer biological networks.

Studying biological networks via experiment is usually costly and time-consuming whereas systems biology that computationally and mathematically models complex biological systems is a complementary approach (e.g. [5], [6], [7]). Mathematical modeling has played an important role in understanding, predicting and even manipulating biological processes. For example, cell-specific responses to the cytokine TGFβ was found determined by variability in protein levels through mathematical modeling [8]. While pluripotent stem cell (PSC) gene regulatory network is largely unknown, a model was developed to predict network states in response to external stimuli [9]. A model was proposed to simulate a vector-borne disease and evaluate the impact of an imperfect vaccine [10].

Building mathematical models for an unknown biological network from experimental data is a type of reverse engineering. Various methods have been developed to infer biological networks (e.g. [11]–[13]). Although remarkable success has been achieved (e.g. [14]–[16]), there are still plenty of issues concerned in applications, including lack of prior knowledge, incomplete data collection, large-scale networks, stability of constructed models, etc. This thesis presents novel methods to infer complex biological networks.

1.1 General structure of biological networks

Nodes and edges are basic elements of a network. Nodes represent units of a network and edges indicate interactions among these units. For a biological network, nodes can denote a wide range of biological units from species in an ecosystem, over genes and proteins of photosynthesis systems of plants, to neurons in the brain. At any given time, nodes are associated with a quantity such as molecular concentration. A network might contain all existing species within a cell, or an organism, or only a subset of those. It depends on decisions on what is important to capture in a particular system. Often, networks are constrained to a relatively small number of species due to the complexity of a system and/or the available data/information about the system. It is common to see gene, protein or metabolic networks, but not all combined. Or one can find detailed networks of a particular (relatively small) cellular process.

Another principle attribute of biological networks is dynamical behavior over time. Direct observations of this feature can be obtained from experiments. For example, the expression level of genes evolves with time. Under this context, edges (links) reveal causal relationship among these nodes, for example, downstream and upstream regulations of transcripts and signal transduction pathways of neurons. Links in causal networks are defined as follows. We say a link from A to B exists if, after removing all other nodes in the network, B at any time depends on past and present values of A. Figure 1.1.1 (left column) shows a simple example of a gene regulatory network containing three genes that are regulated in sequence. Note that one cannot add a link from gene A to gene C if all three genes are considered because, by physically removing gene B, gene A and gene C are disconnected. Nevertheless, if one only constructs a network of gene A and gene C without gene B because, for example, gene B is not measurable, the original network becomes the one in the right column. In this case, the transitivity from gene A to gene C is preserved because they are physically connected via gene B even if gene B is not observable.

A complete knowledge of a biological network consists of two parts: topology (causal relationship among nodes) and internal dynamics (interaction mechanisms between nodes). In graph theory [17], causal network topology can be expressed as a directed graph. The graph is written as an ordered pair $G = (V, E)$ where V is a set of nodes (units) of the network and E is a set of ordered pairs of edges. For example, an edge between nodes x and y expressed as (x, y) represents a cause-effect from x to y. While topology captures the interaction map (like a map of a city), dynamics capture the flow (like the traffic in that city) and how this changes over time. Mathematical models are effective tools to describe internal dynamics. Commonly used models include ordinary differential equation (ODE), stochastic differential equation (SDE) and partial differential equation (PDE) [18].

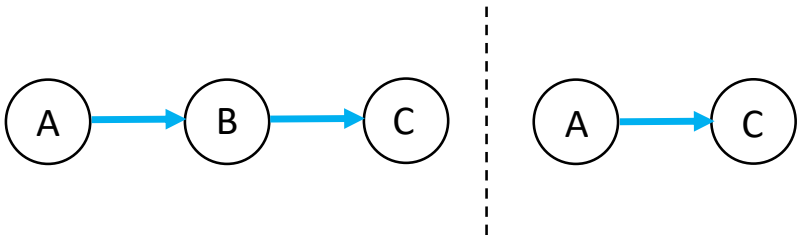


Figure 1.1.1: A gene regulatory network. Left column shows the ground truth network consisting of four genes. Right column shows the network, where gene B is a hidden node but the transitivity from gene A to gene C is preserved.

1.2 Inference methods for biological networks

Topology and dynamics of most biological networks are largely unknown. Network inference aims to learn the topology of the target biological network and construct a mathematical model to represent its internal dynamics. A large number of inference methods have been developed to infer biological networks. However, since most biological networks involve substantial nodes, topology and internal dynamics are not always explored simultaneously. Here, we review some mainstream techniques that are used in practice. A thorough overview of this topic can be found in [19]–[23]. We start by introducing methods that mainly focus on inferring network topology.

1.2.1 Correlation-based methods

Correlation-based methods are widely used to infer associations between nodes [23]. The inference result indicates whether two nodes are related according to some metric. However, it may or may not detail causal relationship. Hence, a biological network is not fully learned with this type of method.

The computational cost of correlation-based methods is cheap and scales well with respect to the number of nodes. Correlation-based methods are usually applied where there are not enough data to support a complete inference of the target network. There are two important classes of correlation-based methods: value-based methods and rank-based methods. Value-based methods include Pearson's correlation [24], [25], distance covariance [26], [27], Theil-Sen estimator [28], [29], and partial correlation and information theory [30], [31].

Pearson's correlation measures the strength of the linear relationship between two random variables and is given by:

$$corr = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)S_x S_y} \tag{1.2.1}$$

where $\{x_i\}$ and $\{y_i\}$ are samples of two random variables, $\bar{x}$ and $\bar{y}$ are sample mean, and S_x and S_y are standard deviations.

Distance covariance examines the statistical dependence of two random variables:

$$dcov(X,Y) = \frac{1}{n^2}\sum_{ij} X_{ij}^c Y_{ij}^c \tag{1.2.2}$$

where

$$X_{ij} = |x_i - x_j|,\ Y_{ij} = |y_i - y_j|$$
$$X_{ij}^c = X_{ij} - \bar{X}_{i,:} - \bar{Y}_{:,j} + \bar{X},\ Y_{ij}^c = Y_{ij} - \bar{Y}_{i,:} - \bar{X}_{:,j} + \bar{Y}$$
$\bar{X},\ \bar{Y}$: grand mean of matrix
$\bar{X}_{i,:},\ \bar{Y}_{i,:}$: row mean of matrix
$\bar{X}_{:,j},\ \bar{Y}_{:,j}$: column mean of matrix

Theil-Sen estimator is a robust way to fit a line to sample points on a plane defined as:

(1.2.3)

$$TS = median\left\{m_{ij} = \frac{y_i - y_j}{x_i - x_j} : x_i \neq x_j, 1 \leq i \leq j \leq n\right\}$$

Partial correlation and information theory applies Data Processing Inequality (DPI) to rule out indirect interactions:

(1.2.4)

$$PCIT_{xy} = \frac{corr(x,y) - corr(x,z)corr(y,z)}{\sqrt{(1 - corr(x,z))^2(1 - corr(y,z))^2}}$$

Rand-based methods are more robust to outliers than value-based methods [23]. Kendall measure is one of the most commonly used methods [32]:

(1.2.5)

$$K(x,y) = \frac{con(x^\tau, y^\tau) - dis(x^\tau, y^\tau)}{0.5n(n-1)}$$

where x^τ and y^τ are the ranked expression profiles of genes x and y, and con and dis denote the number of concordant and disconcordant pairs, respectively.

Other rank-based methods include inner composition alignment (ICA) [33] and Hoeffding's coefficient [34].

1.2.2 Causality-based methods

One of the most well-known causality measures is Wiener-Granger causality (WGC) [35], [36]. It is used to infer causal influence between two variables. Given time series of variables X and Y, if using the past records of both X and Y improves prediction of X using time series of X only, Y is said to be G-cause X. Granger test is a method based on the hypothesis that the underlying system is a linear multivariate stochastic process. It has been widely applied to measure the interactions among subdomains of brain using MEG, SEEG and EEG data [35]. Nevertheless, this method is less effective for nonlinear systems [23].

Convergence cross mapping framework (CCM) is another causality-based method [37], [38]. CCM reconstructs a phase space for each variable using the delay-coordinate embedding method, where the attractor manifolds of phase spaces are used for estimation. The correlation coefficient between the original time series and the estimated ones is a measure of CCM causal influence. The strength of causality is reflected by the value of correlation coefficient. If the coefficient of an ordered pair of variables is non-positive, one variable has no influence on the other. The method requires a considerable amount of data and the system should be chaotic, so that the trajectory takes values close to its past values. Additionally, the method is very sensitive to noise [39].

1.2.3 Information-theoretic based methods

Mutual information (MI) is a basic information-theoretic method to quantify pairwise dependence between two variables. MI indicates how much knowing one variable helps learn the other [40], [41]. The dependence is determined by the similarity between joint distribution and factored distribution:

(1.2.6)

$$MI(x,y) = \sum_x \sum_y p(x,y) log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

where $p(x,y)$ is joint probability distribution of x and y, and $p(\cdot)$ denotes marginal probability distribution. MI measure is symmetric and non-negative. Hence, the obtained networks are undirected. Calculating MI requires estimating the probability distributions from data. Nevertheless, how to attain unbiased MI measure is still an open question [23].

Relevance network approach (RELNET) [42], [43], context likelihood of relatedness (CLR) [44], maximum relevance/minimum redundancy feature selection (MRNET) [45] and accurate reconstruction of cellular networks (ARACNE) [46] are all commonly used MI-based network reconstruction methods. RELNET and CLR methods are not able to distinguish direct and indirect relationships. MRNET and ARACNE attempt to remove indirect relationships. However, MRNET can only achieve a local optimal solution since it applies a greedy algorithm. ARACNE only considers indirect relationships within each triplet and requires a large amount of data [23].

Another information-based method is called maximum information coefficient (MIC) [47], [48]. The principle of MIC is that if a relation exists between two variables, the scatterplot of their data can be partitioned using a grid. For a grid (*x*-by-*y*), a probability distribution induced on each possible box of the grid is constructed, where the probability of a box is proportional to the number of data points falling inside the box. MI is calculated for each box based on the induced distribution. $m_{x,y}$ is the maximum MI over all boxes of the grid (*x*-by-*y*). MIC is the maximum $m_{x,y}$ over all ordered pairs (x,y). However, it was argued that MI-based methods are more suitable than MIC to equitably quantify associations in large datasets [41].

1.2.4 Graphical Gaussian model (GGM)

Graphical Gaussian models generate undirected graphs of biological units [49], [50]. Edges in the graph show pairwise dependence of two nodes conditioned on all the other nodes. Each node in the graph is a random variable and all the nodes are assumed to be jointly Gaussian distributed with 0 mean and covariance matrix C. GGM is constructed based on covariance matrix C. In principle, two nodes are independent conditional on the remaining units if and only if the corresponding element in inverse covariance matrix C^{-1} is zero. To estimate covariance matrix C, nodes are considered as a multivariate Gaussian distribution. In this case, we assume time series data are independently sampled from that distribution. C is approximated by the empirical estimation:

(1.2.7)

$$\hat{C} = \frac{1}{n-1}\sum_{t=1}^{n}(X_t - \bar{X})(X_t - \bar{X})'$$

where X_t is a vector containing the time series of all the nodes at time instance t and $\bar{X}$ is

the mean of the time series.

A variant of GGM is graphical Lasso (Glasso) [51]. The inverse covariance matrix is estimated directly via the maximum a posteriori method (MAP) where a penalty is added to impose sparsity to the matrix:

(1.2.8)

$$C^{-1} = \underset{S}{\operatorname{argmax}} \log\det(S) - trace(\hat{C}S) - \lambda\|S\|_1$$

1.2.5 Bayesian network (BN)

Bayesian networks are a widely applied probabilistic graphical model, which is described as a directed acyclic graph (DAG) [52], [53]. Nodes of the graph are random variables and directed edges present conditional dependence among nodes. Consequently, each node is associated with a probability function that takes a set of parent nodes as conditioned random variables. Exact inference (computing the probabilities exactly) in Bayesian networks is NP-hard [54]. Therefore, in real-world applications, constraints either on topological structure of the graph or conditional probabilities are imposed and approximate inference (computing probabilities with reasonable precision) is conducted (e.g. bounded variance algorithm [55]). Another alternative is to implement a heuristic search of local structure around some variables using algorithms such as children algorithm [56] and Markov blanket algorithm [57].

One advantage of BN is that it allows unobservable nodes, which is very important for studying biological networks as normally not all the biological units can be measured. However, a general BN may not indicate causal relationships among nodes as directed edges are based on conditional dependence. In addition, since a Bayesian network is a DAG, information only flows in one direction and no feedback loops are allowed in the network.

1.2.6 Model-based inference

All the methods introduced above only infer network topology or even just reveal partial information of network structure. To fully understand a biological network, learning internal dynamics is also essential to interpret underlying mechanisms of the network. Model-based methods use dynamical models to describe internal dynamics. Meanwhile, model structure indicates network topology. A dynamical model can not only help understand a biological network but also predict its behavior under various conditions without real experimental observations [58]. Additionally, by comparing dynamical models of a biological system under different conditions (e.g. treatment, mutation, environmental stimuli and location of organisms), one can find systematic changes between these models, thus understanding crucial internal mechanisms. Its power to assist learning biological networks has drawn increasing attention in the field of systems biology. One successful application is on *Arabidopsis* circadian clock [59]–[64].

Although the inference methods above are easy to use and applicable to a wide range of biological networks, they cannot identify system dynamics, thus causing limited performance. In contrast, model-based methods considerably improve inference accuracy by exploring internal dynamics. The trade-off is that model-based methods have higher requirements on data quality (e.g. sampling frequency, information richness, length of time series) and can be computationally demanding. Next, we introduce model-based methods that have been widely

used to infer biological networks.

1). Compressive sensing based methods (CS)

Compressive sensing based methods have been popular in recent years [65], [66]. This paradigm was originally developed to reconstruct signals using limited data. The basic idea is to formulate system identification as a linear regression problem and solve a regularized optimization problem:

(1.2.9)

$$\widehat{w} = \underset{w}{\operatorname{argmax}} \|Y - \Phi w\|_2^2 + \lambda \|w\|_1$$

where Y are outputs of the model (e.g. shifted time series, derivatives), Φ is called dictionary matrix and w contains model parameters. The formulation of Φ highly depends on the postulated model. For example, Φ can be simply a stack of column vectors consisting of the time series of nodes (in case the model is linear) [67], or it can be constructed using nonlinear basis functions (nonlinear models) [68], [19]. There are many other variants of this Lasso-type method, such as elastic net method [69], time-varying sparse regression (Tesla) [70] and sparse Bayesian learning (SBL) [71]. Most compressive sensing based methods require tuning regularization variable λ that controls the trade-off between data-fitting and regularization, which demands extra computation efforts and causes information loss.

2). Hierarchical Bayesian regression model (HBR)

Hierarchical Bayesian regression model uses a linear regression model to describe regulations from other nodes to a specific node [72]. The likelihood of the target node is assumed to be Gaussian:

(1.2.10)

$$p(y|w, \sigma^2, \Phi, \mathrm{k}) = \mathcal{N}(\Phi \mathrm{w}, \sigma^2 I)$$

where y are outputs related to the target node (e.g. shifted time series, derivatives), Φ is a dictionary matrix containing data of the other nodes, σ^2 is noise variance in the model, k is the index for sets that contain regulators of the target node and w contains model parameters. The prior distribution of w is set to be Gaussian as the conjugate of the likelihood:

(1.2.11)

$$p(w|\gamma, \sigma^2) = \mathcal{N}(0, \gamma\sigma^2 I)$$

Gamma distributions are used as conjugate priors for both σ^{-2} and γ^{-1}. A uniform prior is assigned to topology index k.

The resulting posterior distribution for the linear regression model is:

(1.2.12)

$$p(w, \gamma, \sigma^2, k|y, \Phi) \propto p(y|w, \sigma^2, \Phi, \mathrm{k}) p(w|\gamma, \sigma^2) p(\gamma) p(\sigma^2)$$

Finally, Markov chain Monte Carlo method (MCMC) is used to draw samples of all the variables from the posterior distribution (1.2.12). The collected samples can be used to infer network topology and estimate model parameters.

Other variants of HBR include Bayesian spline autoregression (BSA) [73] and non-homogeneous hierarchical Bayesian model [72]. HBR contains a sampling loop in its framework leading to heavy computational burdens. The method is computationally prohibitive for large-

scale networks.

3). Gaussian process methods

Gaussian process is an important tool in non-parametric Bayesian statistics, where prior distributions are defined for functions instead of parameters [74], [75]. Applications of Gaussian process are very flexible and depend on the models proposed to describe a network. Here, we describe one simple case. More details will be discussed in later chapters. Assume the following model is used to describe interactions from other nodes to a specified node:

(1.2.13)

$$y_t = f(x_t) + e_t$$

where y_t are scalar outputs related to the target node (e.g. shifted time series, derivatives), x_t is a vector containing data of a particular set of nodes (indexed by π), e_t is white Gaussian noise with noise variance σ^2 and $f(\cdot)$ is an unknown function.

Assume function $f(\cdot)$ is contained in a predefined functional space that is characterized with kernel function $k(\cdot,\cdot)$. Function $f(\cdot)$ can be interpreted as a Gaussian process with zero mean and covariance matrix K, where $E\left(f(x_i)f(x_j)\right) = [K]_{ij} = k(x_i, x_j; \beta)$ and β is a hyperparameter controlling the properties of the kernel function. It can be shown that the likelihood of the model is a Gaussian distribution. Given data $Y = [y_1, \dots, y_n]'$ and $X = [x_1, \dots, x_n]'$, the likelihood is $p(Y|X, \sigma^2, \beta, \pi) \sim \mathcal{N}(0, \sigma^2 I + K)$. Variables, σ^2 and β can be estimated by maximizing the likelihood function.

Assuming a uniform prior is assigned to π, the conditional posterior distribution of π is:

(1.2.14)

$$p(\pi|Y, X, \sigma^2, \beta) = \frac{p(Y|X, \sigma^2, \beta, \pi)p(\pi)}{\sum_{\Pi} p(Y|X, \sigma^2, \beta, \Pi)p(\Pi)}$$

Moreover, the probability of a specific edge from node j to the target node i is:

(1.2.15)

$$p(j \to i|Y, X, \sigma^2, \beta) = \sum_{\pi} I(j \in \pi)p(\pi|Y, X, \sigma^2, \beta)$$

where $I(\cdot)$ is the indicator function.

Evaluating (1.2.14) and (1.2.15) requires enumerating all possible sets of regulators of the target node (indexed by π). This task is computationally prohibitive for large networks. In practice, a constraint is usually imposed to the cardinality of these sets (e.g. $|\pi| \leq 3$) [76]. Nevertheless, it is possible that the number of links connecting to a node exceeds the maximum cardinality, thus causing systematic errors during inference.

4). Gaussian Bayesian network (BGe)

Gaussian Bayesian networks [77], [78] regard output y_t of the model (related to a specific node) and other nodes (regulating this specific node) at any time instance t as joint Gaussian distributions. Samples at different time instances are assumed to be independent: $p(z_t|\mu, \Sigma, \pi) = \mathcal{N}(\mu, \Sigma)$ and $p(z_1, \dots, z_n|\mu, \Sigma, \pi) = \prod_{t=1}^{n} \mathcal{N}(\mu, \Sigma)$ where $z_t = [y_t, x_t]$, x_t contains the data of a particular set of regulators (nodes) indexed by π, and μ and Σ are unknown mean and covariance matrix, respectively.

A conjugate normal-Wishart prior is imposed to mean μ and precision matrix Σ^{-1} so that the marginal distribution, $p(z_1, \dots, z_n|\pi) = \iint p(z_1, \dots, z_n|\mu, \Sigma, \pi)p(\mu, \Sigma|\pi)d\mu d\Sigma$ can be computed in closed-form [74].

Finally, assuming regulator sets are uniformly distributed, the posterior distribution of regulator set π is:

(1.2.16)

$$p(\pi|z_1, \dots, z_n) = \frac{p(z_1, \dots, z_n|\pi)p(\pi)}{\sum_{\Pi} p(z_1, \dots, z_n|\Pi)p(\Pi)}$$

Similar to (1.2.14), a constraint can be set on the cardinality of set π to avoid the combinatorial search of all possible groups of regulators. This method considers the model as a multivariate Gaussian process. However, correlations among time instances of z_t are simplified to be independent.

5). Mixture Bayesian network

The setting of mixture Bayesian network models [79], [80] is analogous to that of Gaussian Bayesian networks, in that time series are treated as independent samples from a distribution. Instead of using a normal joint Gaussian distribution to describe the relation between the target node and its regulators in BGe, a Gaussian mixture model is adopted where the number of mixture components, N is unknown and needs to be determined.

In addition, parameters of the mixture model including means and covariance matrices are considered deterministic but unknown. For fixed N, these variables are optimized by maximizing the likelihood function. Following that, Bayesian information criterion (BIC) is used to determine the best value of N for each set of regulators. Finally, BIC is employed again to decide the optimal regulator set. Compared with BGe, this method applies a more general stochastic process (mixture Gaussian) to represent the model. Nevertheless, correlations across time instances of z_t are not explored, similar to BGe.

6). Dynamic Bayesian network (DBN)

Dynamic Bayesian networks are the extension of Bayesian networks (BN), where a DBN not only describes dependence among nodes at each time step but also includes transitions across time [81]. In a DBN, each time slice is presented as a normal BN, which is conditionally dependent on the previous ones. The structure and dependence of intra-slice (inside a time slice, i.e. static BN) or inter-slice (between different time slices) subnetworks need not be identical, resulting in a generalized graphical representation for many existing models including hidden Markov models and Kalman filters [81]. Although, in general, a DBN is still a DAG, adding feedback loop into the model is straightforward. For example, if node A_t is affecting node B_t at time t, a feedback can be realized by connecting node B_t to node A_{t+1}.

In practice, modeling with a general DBN is infeasible because its dimension increases considerably with the number of biological units and time instances, and the computational complexity of the conditional probability table becomes prohibitive. Therefore, to reduce computational complexity, one can use an identical structure for each intra-slice subnetwork and make inter-slice subnetworks a Markov chain of low order (e.g. first order Markov leading

to a two-slice temporal Bayes net) [82]. To further simplify the model, one can assume the nodes of a DBN are linearly related, leading to a linear state space model [83]. Hence, when solving network inference, it may be more practical to begin with a concrete dynamical model rather than a general DBN.

1.3 Development of novel model-based methods

A biological network is a complex dynamic system containing a large number of interlocking feedback loops. To fully understand such a complicated network, we need to learn its topology and internal dynamics. One critical assumption about network topology is that a biological network is not fully connected but rather sparse [19]. A biological unit in a network is controlled by not all but only part of the others. Internal dynamics of a network show how nodes interact with each other. In particular, they reveal mechanisms of dynamical behavior in response to entrainment from external signals and enforced systematic or environmental variations such as mutation and treatment. Internal dynamics and input-output relationships of a network are relevant but not identical. For example, two networks are allowed to have the same input-output map but completely different internal dynamical behavior and vice versa [84], [85]. In principle, network topology and internal dynamics must be learned simultaneously because they are intimately related. For example, imposing sparsity constraints on network topology when identifying internal dynamics is crucial in many inference methods such as compressive sensing based methods.

We have introduced some methods that infer connectivity among nodes whereas internal dynamics remain unknown (e.g. correlation-based methods, information-theoretic based methods). Meanwhile, many existing system identification methods focus on identifying input-output dynamics of a system for the purpose of control or prediction. Although these estimated models have small generalization errors, their model structure can be totally different from the target network. For example, prediction error minimization method identifies a model purely based on data-fitting, thus always generating a fully connected network. Therefore, common system identification methods cannot be used directly to infer networks. We have presented several model-based inference methods and discussed their limitations. However, some important issues are not considered in these methods, such as hidden nodes and system stability. In particular, how to deal with hidden nodes whose number and identity are unknown and impose system stability is problematic when inferring biological networks.

This thesis develops novel model-based inference methods to study biological networks so that both internal dynamics and network topology are learned from data. Consequently, internal dynamics are fully interpreted by the established model and network topology is explicitly shown by the model structure. In particular, we target on several problems of interest in network inference, including limited data source, sparsity of network topology, stability of constructed models, existence of unobservable nodes and large-scale networks. These are typical properties we find in systems biology.

1.3.1 General framework

Network inference aims to find a mathematical model whose structure and internal

dynamics are consistent with the target network given measured data. Therefore, the inference procedure mainly consists of five steps: collecting data, selecting a model class, expressing model equations, identifying the proposed model (including model structure and system dynamics) and validating the estimated model.

The quality of data can influence selection of model classes. For example, if only limited data are available, a simple model class may be preferred to avoid over-fitting.

Different models can represent a biological network with different depth. Nonlinear models are most appropriate to extract the nature of biological systems. Linear models, on the other hand, have simpler expressions and are easy to analyze. The selection of model classes depends on various aspects such as prior knowledge of the network and quality of data.

Techniques used to determine model structure and identify system dynamics highly rely on how a model is expressed. If a model is parametrized by variables, the objective is to estimate model parameters. Alternatively, if a model is characterized by system dynamics, functional estimation may be conducted instead (e.g. estimating impulse responses of a linear model). One of the most important issues that must be considered during identification is model parsimony. Model parsimony is related to selection of model complexity, where a selected model must be both descriptively accurate and simple [86]. This results in a trade-off between fitting data and penalizing model complexity. Information criteria including Akaike's information criterion (AIC), Bayesian information criterion (BIC) and minimum description length (MDL) are commonly used to promote model parsimony. There are many other methods that apply relaxations to information criteria and solve model selection problems via numerical optimization (e.g. compressive sensing based methods).

Finally, it is important to validate the estimated model based on simulation or prediction on a new dataset that is not used for identification.

1.3.2 Mathematical models to describe biological networks

Models used to describe biological networks can be classified in two categories: grey-box models and black-box models [87], [88]. Grey-box models are postulated according to physical or biological laws [18]. Construction of these models requires prior knowledge of the target network, including causality among certain biological units and types of biochemical reactions involved. Hence, grey-box models have explicit biological explanations. In contrast, black-box models are postulated in terms of inputs and outputs without any knowledge of their internal operations. Prior knowledge of a network is not necessary to propose a black-box model. Linear models are typical black-box models and widely applied in systems biology. Unlike grey-box models, internal dynamics of black-box models may not have biological interpretations.

This thesis considers both grey-box and black-box models. If the working mechanism of a biological network is well known, one can use grey-box models. For example, a gene regulatory network involves interactions among genes and proteins so Hill functions and Michaelis-Menten kinetics can be used to construct grey-box models [18]. If little or no prior knowledge is available and unobservable nodes exist, one can use black-box linear models to describe the target network. The main linear models we consider include state space model, Output-Error model (OE), autoregressive model with exogenous inputs (ARX) and dynamical structure function (DSF). State space models generalize all linear and nonlinear systems. Identifying a state space model requires full state measurements, which are not available in many real-

world applications. OE models have a simple model structure, thus easy to identify. However, they ignore process noise in biological systems. Identification of ARX models can be formulated as a linear regression problem so a variety of methods are available to promote sparse topology. Nevertheless, applications of linear ARX is limited in 'model power' since only a few networks can be well-approximated by linear ARX in practice whereas construction of nonlinear ARX requires prior knowledge and full state measurements. DSF is a very general linear model but some constraints on its model structure must be imposed due to theoretical limitations.

1.3.3 Novel methods for network inference

This thesis develops novel methods to infer biological networks. Parametric and non-parametric approaches will both be considered. We will specially focus on how to impose sparsity for network topology and system stability.

We formulate the system identification problem in the Bayesian framework to capture stochastic properties of biological networks (including intrinsic, extrinsic and measurement noise). Prior distributions are proposed to encourage sparse network topology or to enforce system stability. Bayesian techniques are applied to tackle resulting intractable probabilistic models. In particular, deterministic approximations (e.g. empirical Bayes) and stochastic approximations (e.g. Markov chain Monte Carlo sampling) form the backbone of the proposed methods.

Consequently, four inference methods are discussed in this thesis, including one-to-one, combine group and element sparse Bayesian learning (GESBL), the kernel method and reversible jump Markov chain Monte Carlo method (RJMCMC). One-to-one is inspired by the traditional pairwise inference of correlation-based methods, where a dynamical system is estimated for each ordered pair of nodes. Since this method is heuristic and intuitive, we only discuss it in a short chapter. The method will be compared with other methods through simulation. GESBL is developed to impose sparse network topology. Unlike pairwise identification of one-to-one, GESBL infers the entire network in one shot. The novelty of GESBL is that it imposes both element and group sparsity (i.e. model complexity and network sparsity) at the same time. The kernel method is introduced to identify DSFs that are proposed to deal with unobservable nodes. The main contribution is focused on problem formulation where identification of DSFs is greatly simplified and the kernel method can be applied to impose system stability. The limitations of DSFs are also discussed. In addition, inference with measurement noise is considered, which has a huge impact on network sparsity but is not specifically discussed in previous research. Finally, RJMCMC approach further improves the kernel method. RJMCMC provides a more effective way to determine model structure by encouraging a global search of network topology. Moreover, it offers an accurate evaluation of inference confidence.

1.3.4 Performance criteria for inference results

If the ground truth of the target network is known, we can compare the estimated network with the true one. We mainly adopt three performance criteria to evaluate inference results. True Positive Rate (TPR) equates to the percentage of the true links in the ground truth network that are identified in the inferred network. False Positive Rate (FPR) is equal to the

percentage of the null links in the ground truth network that are identified incorrectly. Precision (Prec) is the fraction of the number of correct true links over the total number of true links in the inferred network.

TPR indicates how many true links in the ground truth network are captured, which determines whether the inference result provides rich information. Low TPR suggests many true links are missed. FPR implies the amount of inference error. High FPR shows many true links in the inferred network are false. Prec reveals reliability of the inferred network. If Prec is low, correct and wrong true links are not distinguishable in the inferred network. To conclude, a good inference result should have high TPR and Prec, and low FPR.

1.3.5 Validation of the proposed methods via Monte Carlo simulations

Monte Carlo simulations are conducted to validate our proposed inference methods. First, we apply our methods to infer networks that fall exactly into the proposed model class. These models are generated randomly. Simulations are implemented under different conditions to investigate the critical factors that can influence the algorithm performance.

To evaluate the performance of the proposed methods when inferring real biological networks, we take circadian clock of *Arabidopsis* as a case study. The ground truth network is represented by synthesized circadian models (Millar models [61], [62]). These models are highly nonlinear and have analogous dynamics to the real circadian system. Simulated data are collected under different conditions. The proposed methods are also compared with state-of-the-art methods.

1.4 Thesis overview

This thesis focuses on developing novel methods to infer sparse biological networks. These methods are discussed in sequence as complexity of the postulated models grows. Both parametric and non-parametric identification approaches are considered. Overall, we discuss four inference approaches called one-to-one method, combined group and element sparse Bayesian learning (GESBL), the kernel method and reversible jump Markov chain Monte Carlo method (RJMCMC).

We take the circadian clock of *Arabidopsis* as a case study. We validate the proposed inference methods on synthesized circadian models and test their performance under different experimental conditions. These methods are also compared with state-of-the-art methods. We then apply the best methods to infer the circadian Ca^{2+} signaling network using real experimental data. The findings of the signaling network further demonstrate the effectiveness of the proposed methods.

Next is an outline of each chapter. Chapter 2 introduces the main mathematical machinery to model biological systems, including dynamic models for basic biochemical reactions and system identification techniques employed to estimate those models.

From chapter 3 to chapter 6, we discuss different methods to infer sparse networks. Chapters 3 and 4 are parametric identification methods while chapters 5 and 6 are non-parametric. In addition, chapters 4 and 5 apply Bayesian deterministic approximation techniques while chapter 6 is equipped with stochastic approximations. Overall, chapter 3 reviews one-to-one for further comparison. Chapter 5 mainly applies the existing kernel

methods to identify the proposed DSFs. Chapter 4 and 6 present novel inference methods GESBL and RJMCMC.

We begin with a heuristic inference method called one-to-one in chapter 3. Networks consisting of chain links are considered. As a complement to correlation-based methods, a dynamic system is estimated for each ordered pair of nodes. The method can be applied to infer large-scale networks.

Chapter 4 introduces system identification under the Bayesian framework. It also discusses deterministic and stochastic approximations employed as two major techniques in this project. This chapter applies ARX and nonlinear ARX models to describe sparse networks. Group and element sparse Bayesian learning are combined (GESBL) to impose sparse topology and model parsimony.

Chapter 5 adopts a general linear model termed DSF to describe the target network. This model encodes dynamics of hidden nodes whose measurement is not available. The kernel method endowed with empirical Bayes is applied for identification, avoiding the challenging problem of model selection. In this case, stability of the estimated system is guaranteed.

Chapter 6 establishes a full Bayesian model based on DSF, where network topology is treated as a random quantity. A sampling method called RJMCMC is applied to explore the Bayesian model. RJMCMC encourages a global search of the optimal topology as well as unknown variables. In addition, the full Bayesian model enables evaluation of inference reliability.

Chapter 7 validates all the methods by inferring the synthesized circadian systems. Millar models are chosen as dynamical networks analogous to the real circadian clock. Through Monte Carlo simulations, the proposed methods are also compared with state-of-the-art methods under different experimental conditions, including light transitions and sampling frequency.

Chapter 8 apples the best methods according to the previous simulations to infer the circadian Ca^{2+} signaling network using real experimental data. Without prior knowledge, linear models are used to describe the network. The signaling network under different light conditions is inferred independently. The inferred networks are compared to locate the key clock genes that are responsible for the interaction between the circadian clock and $[Ca^{2+}]_{cyt}$ in different light pathways.

Chapter 9 concludes the thesis and discusses the future research.

Chapter 2.

Methodology

Much research has been conducted to learn about biological networks such as circadian clocks in plants. These networks consist of complex dynamics including interlocking loops among network components such as genes and proteins. Studying such networks through experimentation is normally time-consuming and expensive. Mathematical modeling has been a prevalent way to understand causal relationships between species and internal dynamics of biological networks. With the models, one can analyze intrinsic attributes of a network, predict network behavior under various conditions and compare dynamics of different networks.

Building models from data consists of two main steps: selection of model type and system identification from data. Dynamical systems theory provides various types of mathematical models, each with unique properties. Different combinations of these properties lead to different systems, such as nonlinear time-varying systems, nonlinear time-invariant systems and linear time-invariant (LTI) systems.

One can use nonlinear systems for modeling. The model structure is postulated to be consistent with the natural attributes of the target biological system while the model parameters are unknown (grey-box models). Describing biological systems with grey-box nonlinear models involves partial or complete knowledge of the regulatory topology and regulatory mechanisms (types of dynamical functions). In contrast, no prior knowledge and assumption are required to set up linear models (black-box models). However, these models may not have physical interpretations.

After choosing a model class, the next step is to identify the system using measured data. There is a wide range of system identification techniques available, such as Predication Error Minimization (PEM), Maximum Likelihood (ML), Instrumental Variable (IV) [89], Linear Regression, Kernel methods [40], [90], [91] and Sparse Bayesian Learning (SBL) [68], [71]. Some of them are only applicable for linear models while others have wider applicability. While these methods are highly popular and successful in different areas of applications, they also have limitations in network inference. For example, PEM is not able to explore network topology and SBL cannot impose system stability. Hence, this project contributes to improving the existing methods and developing novel ones.

Next is a summary of this chapter. Section 2.1 reviews fundamental concepts of dynamic

systems relevant to this work. Section 2.2 briefly discusses several basic mathematical equations that naturally arise in biochemical reactions. Section 2.3 discusses the basic principle of system identification. Finally, section 2.4 concludes the chapter and emphasizes the major problems to be solved.

The notation of this chapter is standard and will be used throughout this thesis. I denotes the identity matrix. For $L \in \mathbb{R}^{n\times n}$, $diag\{L\}$ denotes a vector which consists of diagonal elements of matrix L and $[L]_{ij}$ presents the ijth entry. $blkdiag\{L_1, \dots, L_n\}$ is a block diagonal matrix. $trace\{L\}$ denotes the trace of the matrix. $L \succcurlyeq 0$ means L is positive semi-definite. $\|w\|^2_{L^{-1}}$ represents $w^T L^{-1} w$. For $l \in \mathbb{R}^n$, $diag\{l\}$ denotes a diagonal matrix whose diagonal elements come from vector l. $[l]_{ij}$ denotes the jth element of the ith group of l. $l \geq 0$ means each element of the vector is non-negative. $v = l^k$ is also a vector where $v_i = (l_i)^k$. $vec\{x_1, .., x_n\} = [x_1, \dots, x_n]'$ means to vectorise elements $\{x_1, .., x_n\}$. A vector $y(t_1 : t_2)$ or $y_{t_1:t_2}$ denotes a row vector $[y(t_1) \quad y(t_1+1) \quad \cdots \quad y(t_2)]$. $\mathcal{N}(w|\mu, \Sigma)$ denotes a Gaussian distribution of w with mean μ and covariance Σ.

2.1 Linear and nonlinear dynamic systems

One of the most important steps in modeling, if not the most important, is to choose the model class. On one hand, the model class has to be rich enough to capture key dynamical behavior in the data. On the other hand, complex models may lead to over-fitting and are difficult to identify. This thesis is focused on time-invariant nonlinear or linear systems. For both simplicity of explaining the methods and to reduce computational complexity, most of the key developments are done for linear systems. We then explain how they can be extended to nonlinear systems.

The standard form of a continuous time-invariant nonlinear system in state space form is expressed as:

(2.1.1)

$$\begin{aligned} \dot{x}(t) &= f\big(x(t), u(t)\big) \\ y &= h\big(x(t), u(t)\big) \\ x \in \mathbb{R}^n, u \in \mathbb{R}^m, y &\in \mathbb{R}^p \end{aligned}$$

where $x(t)$ denote the state variables, $u(t)$ are the inputs to the system and $y(t)$ are the measured outputs. This form of a nonlinear system is quite general, the system structure is very flexible and it can describe complex dynamics. Although nonlinear systems are suitable to describe complex biochemical reaction networks, their identification can require rich datasets and high computational power. Given that a large amount of experimental data is very limited, we introduce a simpler model class known as linear systems.

The standard form of a continuous time-invariant linear system in state space form is:

(2.1.2)

$$\begin{aligned} \dot{x}(t) &= Fx(t) + Gu(t) \\ y(t) &= Hx(t) + Iu(t) \\ x \in \mathbb{R}^n, u \in \mathbb{R}^m, y &\in \mathbb{R}^p \end{aligned}$$

where F, G, H and I are system matrices. When modeling complex nonlinear systems, linear systems can be thought as linearization around some state (assuming the original

nonlinear system is differentiable at that state). Then, trajectories near the linearized state can be good approximations of the original nonlinear system. If the linearized state is an equilibrium point (x_e, u_e), then the nonlinear system (2.1.1) satisfies $f(x_e, u_e) = 0$, with $y_e = h(x_e, u_e)$. In this case, the linearized system around this equilibrium is:

(2.1.3)

$$\dot{\delta x} = \frac{\partial f}{\partial x}(x_e, u_e)\delta x + \frac{\partial f}{\partial u}(x_e, u_e)\delta u$$
$$\delta y = \frac{\partial h}{\partial x}(x_e, u_e)\delta x + \frac{\partial h}{\partial u}(x_e, u_e)\delta u$$
$$\delta x = x - x_e,\ \delta y = y - y_e$$

Most biochemical systems will have intrinsic, extrinsic and measurement noise. Noisy linear systems can be expressed as stochastic differential equations (SDE) [92], [93]:

(2.1.4)

$$dx(t) = [Fx(t) + Gu(t)]dt + U\Lambda^{\frac{1}{2}}dW(t)$$
$$y(t) = Hx(t) + Iu(t) + v(t)$$
$$x \in \mathbb{R}^n, \bar{w} \in \mathbb{R}^n, u \in \mathbb{R}^m, y \in \mathbb{R}^p, \bar{v} \in \mathbb{R}^p$$

where $dW(t)$ is called process noise and is the increment of a Wiener process, and $v(t)$ is measurement noise:

(2.1.5)

$$dW(t) = W(t + dt) - W(t) \sim \mathcal{N}(0, dt)$$

The stochastic dynamic model in (2.1.4) is just one model of noise. More complicated models are discussed in [92].

The goal of modeling is to find the model parameters in the above equations from both prior knowledge and data. A common tool in system identification is called prediction error minimization. It is an indirect method since it first estimates a discrete-time system using sampled input-output data and then converts the system back to continuous-time [58], [94]. To discretize the continuous-time system in (2.1.4), assume the input u takes piecewise constant values (Zero Order Hold) within a sampling period. The resultant discrete-time linear model corresponding to (2.1.4) is [92]:

(2.1.6)

$$x(k+1) = Ax(k) + Bu(k) + w(k)$$
$$y(k) = Cx(k) + Du(k) + v(k)$$

where

$$A = e^{FT},\ B = \int_0^T e^{F\tau}d\tau G,\ C = H,\ D = I$$

$$E\{w(k)\} = 0, cov\{w(k), w(l)\} = Q\delta(k - l)$$

$$Q = \int_0^T e^{F\tau}\tilde{Q}e^{F^T\tau}d\tau,\ \tilde{Q} = U\Lambda U$$

$\delta(k)$: Kronecker delta function

T: sampling period

State space models are just one way to describe dynamical systems. In practice, the choice of mathematical models is flexible and depends on other aspects. For example, if not all the states can be measured, the system may not be identifiable since there may exist multiple realizations that explain the same input-output data. In this case, dynamical structure function (DSF) would describe a dynamical network by encoding hidden states as transfer functions. Hence, this thesis is not constrained to state space models. Rather, we will consider several model types including auto-regression with exogenous inputs model (ARX) and DSF. We will also develop corresponding identification methods to identify these models.

2.2 Mathematical modeling of biochemical reactions

2.2.1 Stochastic systems of biochemical reactions

There are two approaches widely used to describe the time evolution of a biochemical system. One is a deterministic approach, which treats a reaction process to be continuous and completely predictable. The system is described by a set of ordinary differential equations (e.g. reaction-rate equations described by mass action kinetics). The other method regards the process as stochastic due to inherent microscopic random molecular fluctuations. The stochastic model uses random-walk processes, governed by differential equations termed master equation to present the dynamics [95]. It is argued that a stochastic formulation has much firmer biological basis than the deterministic one [95]. However, master equations are usually mathematically intractable (problem of moment closure). Numeric stochastic simulation algorithms are typically used to simulate the chemical master equation. Such algorithms provide an equivalent realization of a stochastic model [95].

We start by describing stochastic systems as these are more general, at the cost of increasing complexity in modeling, analysis and simulation. Stochastic systems can be simplified to deterministic systems with process noise, also known as stochastic differential equations (which we consider in this thesis). This subsection reviews some results from the literature on how to simplify stochastic systems.

Suppose the thermally equilibrized mixture of chemical species S_i $(i = 1, \dots, N)$ with its molecular population or number of molecules denoted by $X_i(t)$ $(positive\ integer)$ is contained in volume V. These *N* species interreact through *M* chemical channels R_μ $(\mu = 1, \dots, M)$. The stochastic reaction constant, c_μ which characterizes physical properties of molecules and the temperature of the system is then defined to describe the chemical kinetics such that:

(2.2.1)

$$c_\mu dt = \textit{average probability of a combination of reactant molecules in channel } R_\mu \textit{ will react within the next infinitesimal time interval } dt$$

This constant is also connected with the deterministic reaction-rate constant [95].

The stochastic reaction constant is used to derive various stochastic models to describe a biochemical system. One exact consequence of it is the master equation which is a traditional method to present the time evolution of a chemical reaction [96]:

(2.2.2)

$$\frac{d}{dt}P(x,t|x_0,t_0)=\sum_{j=1}^{M}\left[a_j(x-v_j)P(x-v_j,t|x_0,t_0)-a_j(x)P(x,t|x_0,t_0)\right]$$

where

$x=(X_1,\dots,X_N)^T$: state vector denoting the molecular population of species
a_j: propensity function
v_j: state change vector
v_{ji}: change in the number of S_i molecules due to R_j reaction
$P(x,t|x_0,t_0)$: conditional probability of $x(t)$ given $x(t_0)$

The master equation implies the time evolution of a jump-type Markov process of a state vector. If it can be solved, then the statistical property of a chemical process is fully known. However, the solution is normally mathematically intractable.

Another consequence of (2.2.1) is the next-reaction density function [96]:

(2.2.3)

$$p(\tau,j|x,t)=a_j(x)\exp\left(\sum_{k=1}^{M}a_k(x)\tau\right)$$

$$0\le\tau\le\infty;j=1,\dots,M$$

where $p(\tau,j|x,t)$ denotes the probability that next reaction R_j will occur in the infinitesimal time interval $[t+\tau,t+\tau+d\tau)$ given $x(t)$.

The next-reaction density function is the basis of stochastic simulation algorithms which construct realizations of the state vector and such realizations are consistent with the master equation [96]. Other by-products of the master equation include chemical Kramers-Moyal equation and Foker-Plank equation.

We can calculate the expected value of $x_i(t)$ by multiplying the master equation in (2.2.2) by $x_i(t)$ and sum over x. We would then obtain an equivalent set of differential equations given by:

(2.2.4)

$$\frac{d}{dt}E(x_i)=\sum_{j=1}^{M}v_{ij}E[a_j(x)]$$

where $E(\cdot)$ denotes the expected value. As explained in [12], “whenever fluctuations are not important, the species evolve deterministically according to”

(2.2.5)

$$\frac{dx_i}{dt}=\sum_{j=1}^{M}v_{ij}a_j(x)$$

This deterministic equation, known as reaction rate equation of conventional chemical kinetics, is a good approximation of the original master equation provided that the populations of all species are very large compared to 1. The systems and data we analyze in this thesis satisfy this assumption to some extent. To include some level of intrinsic noise, and also extrinsic noise, we next introduce stochastic differential equations, which is the model we will use throughout the thesis. This choice of model class strikes a balance between accuracy and

complexity.

The stochastic differential equation derived from the mater equation in (2.2.6) is known as the chemical Langevin equation. This equation evaluates the uncertainty of a chemical reaction with considerably less complexity than the master equation, by regarding the chemical system as a continuous Markov process. Unlike the master equation, Langevin equation assumes the molecular population to be real number instead of positive integer [97]:

(2.2.6)

$$dx_i(t) = \sum_{j=1}^{M} v_{ji} a_j(x(t))dt + \sum_{j=1}^{M} v_{ji} a_j\big(x(t)\big)^{\frac{1}{2}} N_j(0,1)\, dt^{1/2}$$

where $N_j(0,1)$ are independent Gaussian random variables with mean 0 and variance 1. The term, $dt^{1/2}$ in the equation indicates $x_i(t)$ is non-differentiable. However, for heuristic reasons, it is often assumed that its derivative exists [97]. As a result, a more convenient expression of the Langevin equation is:

(2.2.7)

$$\frac{d}{dt}x_i(t) = \sum_{j=1}^{M} v_{ji} a_j(x(t))dt + \sum_{j=1}^{M} v_{ji} a_j\big(x(t)\big)^{\frac{1}{2}} \Gamma_j(t)$$

$$\Gamma_j(t) = \lim_{dt\to 0} N_j(0, \frac{1}{dt})$$

$$cov\Big(\Gamma_j(t), \Gamma_j(s)\Big) = \delta(t-s)\ (Dirac\ delta\ function)$$

$$\delta(0)dt = 1 \Rightarrow \frac{1}{dt} = \delta(0)$$

where $\Gamma_j(t)$ are independent white Gaussian noise.

The Langevin equation separates a chemical reaction into two parts. The first term is the deterministic component of the differential equation, while the second term represents the random process originating from molecular random fluctuations (including both intrinsic and extrinsic noise). The Langevin equation belongs to the class of SDE, which supports mathematical modeling using general linear or nonlinear SDE. Figure 2.2.1 summarizes the classes of systems discussed above.

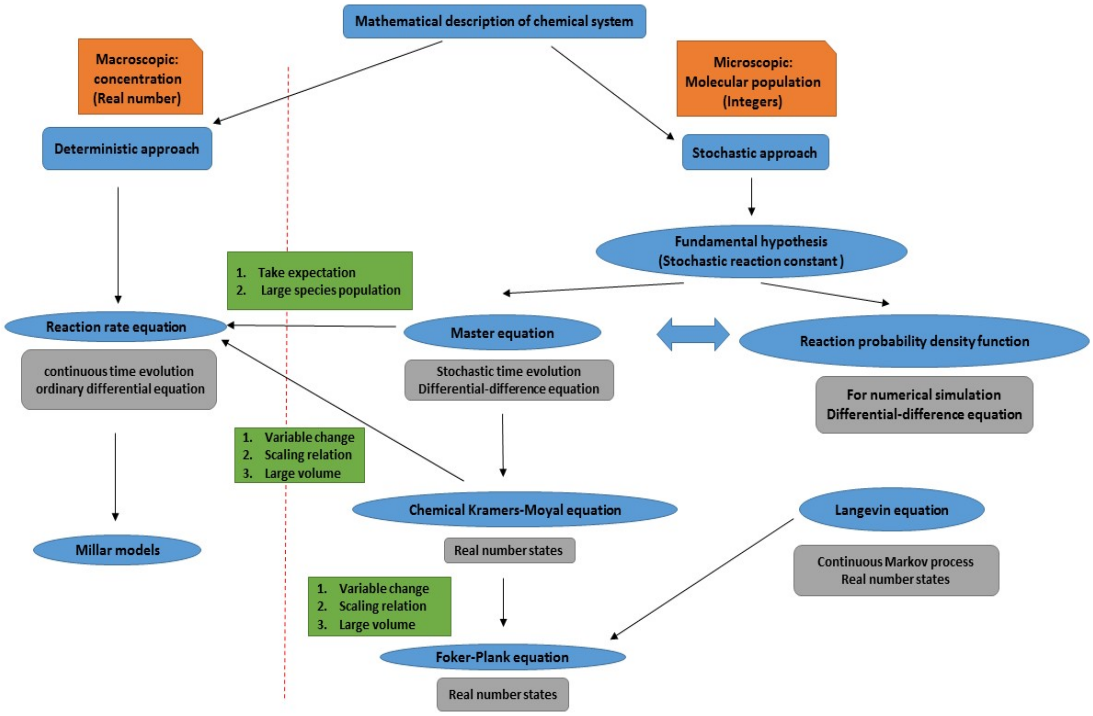


Figure 2.2.1: Hierarchical network of mathematical modeling of biochemical reaction. Modeling is classified into two categories: deterministic and stochastic. They each have different kinds of models to describe the biochemical reaction either on a macroscopic or microscopic level. Grey boxes highlight main properties of the corresponding model. Green boxes show major steps of manipulation of transformation.

2.2.2 Enzyme kinetics of biological networks

Biological networks are composed of a large number of biochemical reactions. Many of these reactions depend on proteins termed enzymes, which play a critical role in the catalysis of metabolites. While differential equation models were initially used to model enzyme kinetics, they are now used to capture dynamics of most biochemical systems. Since these equations play such a central role in biochemical modeling, next we review their derivation.

In enzyme kinetics, substrates are catalyzed into products. Models of these products have been derived from basic chemical mass action kinetics [18], [19]. Assume that a substrate S reacts with an enzyme *E* to form a complex SE. This complex is then converted to a product P plus the enzyme, which is now free to react with another substrate. The entire process is expressed as follows [98]:

(2.2.8)

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} SE \xrightarrow{k_2} P + E$$

where k_1, k_{-1} and k_2 are constants presenting the reaction rate.

The law of mass action indicates the rate of a reaction is proportional to the product of reactant concentrations. Applying the law to the expression (2.2.8) leads to a series of ordinary differential equations (ODE):

(2.2.9)

$$\frac{ds}{dt} = -k_1 es + k_{-1} c$$
$$\frac{de}{dt} = -k_1 es + (k_{-1} + k_2) c$$
$$\frac{dc}{dt} = k_1 es - (k_{-1} + k_2) c$$
$$\frac{dp}{dt} = k_2 c$$

where s, e, c and p denote the concentration of S, E, SE and P, respectively. Note that since enzyme E is only a catalyst which facilitates the reaction, its total concentration is conserved. This feature of the enzyme is also reflected from the equation (2.2.10):

(2.2.10)

$$\frac{dc}{dt} + \frac{de}{dt} = 0 \implies c + e = e_0 = cont$$

Substituting (2.2.10) into (2.2.9) simplifies the model to three ODE equations:

(2.2.11)

$$\frac{ds}{dt} = -k_1 e_0 s + (k_{-1} + k_1 s) c$$
$$\frac{dc}{dt} = k_1 e_0 s - (k_{-1} + k_1 s + k_2) c$$
$$\frac{dp}{dt} = k_2 c$$

The ODEs in (2.2.11) are solved by assuming the formation of complex SE is very fast, after which its equilibrium of the reaction can be reached. This is also known as the quasi-steady state approximation [99]. Such assumption is reasonable if the concentration of the enzyme is small compared with the substrate [98]:

(2.2.12)

$$\frac{dc}{dt} = 0$$
$$c(t) = \frac{e_0 s(t)}{s(t) + K_m}$$
$$v = -\frac{ds}{dt} = \frac{dp}{dt} = \frac{k_2 e_0 s}{s + K_m}$$

where $K_m = \frac{k_{-1} + k_2}{k_1}$ is called the Michaelis constant. Equation (2.2.12) is known as Michaelis-Menten kinetics.

The above example discusses the case where the enzyme has one binding site so that it can only combine with one substrate molecule. However, in practice, an enzyme can have multiple binding sites. A reaction is called cooperative if an enzyme is able to bind more than one substrate molecules. One feature of cooperative reactions is that the binding of one substrate molecule at one site can influence the following binding of other substrate molecules to the remaining binding sites and this behavior is called allosteric effect [18]. If a substrate after binding to one site promotes the binding activity at another site, this substrate is known as an activator. Otherwise, it is an inhibitor. This phenomena is described by another function called

the Hill function [98]:

$$V = \frac{Qs^n}{K_m + s^n} \tag{2.2.13}$$

where V is reaction velocity, Q and K_m are constants, and n is positive but not necessarily an integer.

A reaction can have positive, zero or negative cooperativity if $n < 1, n = 1$ or $n > 1$, respectively. Cooperativity indicates whether a substrate is an activator or inhibitor.

Michaelis-Menten kinetics and Hill functions are currently used not only to model enzyme kinetics, but also general biochemical networks on a macroscopic level. They have proved quite effective in building models such as the Millar models in [59]–[61], [63], [100] describing the circadian system of *Arabidopsis*. In those models, degradation of mRNA and proteins is described by Michaelis-Menten functions while transcriptional activities (activation or inhibition) are presented by Hill functions [58]. Reaction rates of translation processes are directly proportional to concentrations of mRNAs.

Mathematical models introduced to describe biochemical reactions in this section are established strictly on physical laws. These models fully interpret the network topology and internal dynamics of a biological network. Constructing such complex nonlinear models requires comprehensive understanding of the target network. As the target network is mostly unknown in practice, assumptions are often made according to experimental observations as a compromise. For example, the model structure of Millar models is fixed *a priori* during inference. Only model parameters that control the property of biochemical processes are estimated from data. Models constructed in this way can normally predict new experimental observations accurately, because model parameters are estimated to fit a large amount of data. Nevertheless, the network topology indicated by model structure may not be consistent with the ground truth. In particular, as the size of the network grows, updating model structure based on subjective hypothesis becomes prohibitive and the risk of introducing biased information increases.

To infer large-scale networks, it is essential to reduce the reliance on prior knowledge and assumptions. Embedding detection of network topology (i.e. determination of model structure) into the inference procedure can effectively avoid biased result and produce reliable estimation. In order to do that, a compromise is made between model complexity and accuracy. One method is to approximate complex nonlinear functions by pre-fixed basis functions. Model structure is then determined by selecting basis functions (e.g. [76], [68]). The demand of prior knowledge is maximally reduced by using black-box models. Black-box models give no insight on the connectivity between internal nodes. However, as a return, black-box nonlinear models are allowed to have much higher model complexity, thus representing dynamics of the target network more accurately. With a high degree of freedom of model structure, black-box nonlinear models are difficult to identify. By linearization (discussed in section 2.1), these models are further simplified by linear systems. Linear models have decent model structure. Therefore, it is more convenient to deal with hidden nodes, impose sparsity and promote system stability using linear models. In addition, linear models are easier to analyze and compare. Early work using linear models achieved great success [101]–[103]. Biologically, a linear model is valid if the experiment is designed so that highly nonlinear

transitions are avoided and all the activities stay in a linear regime [103].

This thesis focuses on linear models and grey-box nonlinear models in which a dictionary of nonlinear functions (e.g. Michaelis-Menten kinetics and Hill function) is used to formulate the model. Internal dynamics and network topology are both inferred from the data with limited prior knowledge and mild assumptions. In what follows, we discuss the general framework to identify these models.

2.3 System identification

System identification consists of three stages: collecting data, selecting a model class, and expressing and identifying the proposed model. A thorough treatment of system identification can be found in [88], [89], [104], [105]. For biological systems, experimental data can come, for example, from microarrays or sequencing data. In practice, it is common that some species (e.g. proteins) of a biological network are not measured due to high experimental cost. As a result, only the expression level of genes (concentrations of mRNAs) is available, which is called incomplete data scenario [21].

Various types of models have been discussed in the last section, which can be used to model a biological network. The selection of model classes highly depends on practical conditions, including quality of data, computational power and prior knowledge. For example, if the data scenario is incomplete, linear models are more suitable to tackle non-measured species. Also, if the source of data is limited, a simple model class is preferred to avoid over-fitting. Overall, one must combine all factors to strike a balance between model complexity and inference accuracy.

Once a model class is chosen, the model can be expressed in different ways. System identification methods are used to estimate the proposed model. Based on the type of model expressions, identification methods are divided into two classes: parametric and non-parametric. For parametric identification, the goal is to determine unknown system parameters [89]. For non-parametric identification, system dynamics are estimated (e.g. impulse responses of a linear system) [88].

2.3.1 Experiment design

According to the theory of system identification, the input-output data recorded from the conducted experiment have a critical impact on the accuracy of identified models. To produce maximally informative data, the system must be sufficiently excited. In other words, the underlying dynamical behavior of a system should be fully activated, given the designed input. Such inputs are called persistently exciting [89].

The richness of dynamical properties of linear models can be reflected on the frequency domain. The frequency content of a linear model is mainly presented within a certain frequency band. Hence, to sufficiently excite a linear model, the frequency bandwidth of the input signal must at least cover the bandwidth of the system. Input signals that contain a wide range of bandwidth include step signals and impulse signals. Furthermore, an input must be able to distinguish different models in a model class. Roughly speaking, given that input, the frequency content of the outputs of any two different models is not identical.

The discussion of persistently exciting inputs for nonlinear systems is much more complex.

Intuitively, an input signal must activate every node of the network for at least a certain amount of time across the whole experiment so that all nodes contribute to the response of the system to the input.

However, biological experimental data often do not fulfill this requirement, either because of constraints on experimental conditions or intrinsic properties of a biological system. For example, light conditions of an experiment cannot be freely adjusted. The spectrum of circadian clock genes tends to concentrate around the oscillation frequency. As a result, estimated models are only reliable within a narrow frequency domain.

2.3.2 Identifiability of models

Identifiability is a central concept in identification problems. Given a persistently exciting input, identifiability is related to the question: whether different values of model parameters lead to equal models [89]. If the input-output property of the ground truth system can be equally represented by different models in the same model class, the identification problem is ill-conditioned because the true system is not identifiable. A model structure may not be identifiable at all values of model parameters. For example, a SISO second order linear system is not identifiable if pole-zero cancellation exists. There are no general rules to guarantee identifiability of all models. Hence, identifiability is discussed in terms of different model structures. Normally, constraints are required to guarantee the identifiability of a model structure and these constraints become stricter for complex models. For example, multivariable ARX models are globally identifiable whereas some conditions on coprime factorization of system matrices must be satisfied for ARMAX models [89]. We will discuss this issue in the following chapters with respect to particular model classes used to describe the target network.

2.3.3 Model selection

In practical applications, the selection of model classes depends on many aspects, including prior knowledge, engineering intuition and data. It is common that the true model class is unknown. Often, models are selected from the combination of multiple model classes. Even if the model class is known *a priori*, one may not adopt this model class for identification due to the quality of data. In practice, experimental data are limited. A complex model fits better the training data than a simple one. However, if the proposed model is over complicated, the model will fit the noise in the data, thus leading to poor prediction of new observations. Hence, the estimated model does not present the key dynamics of the system. Normally, an upper bound for the number of parameters in a model is 3 to 5 times less than the number of data points [90]. This over-fitting problem is commonly encountered in real-world applications. To avoid this problem, model complexity must be penalized (model parsimony). Information criteria such as AIC, BIC and MDL are widely used for this purpose. Other approaches apply smooth penalties for model parsimony. Introducing penalties causes a trade-off between bias and variance of estimation. A strong penalty can dramatically reduce estimation variance but also increase bias, on the other hand [106].

2.3.4 Identification methods

A variety of system identification techniques have been developed for diverse model classes.

We briefly introduce some of them that are intimately related to the proposed methods in this thesis. More details will be discussed in the following chapters.

1). Maximum Likelihood (ML) and Maximum A Posteriori (MAP)

ML and MAP are basic identification methods. They are widely applied in practice. ML treats model parameters as deterministic but unknown variables and estimates them by maximizing the likelihood function of the proposed model so that the resulting model is mostly likely to produce the observed data [89]. MAP follows the Bayesian framework which regards model parameters as random variables and introduces prior distributions to reflect initial belief. For example, priors can be designed for the purpose of model parsimony. Consequently, MAP maximizes the posterior distribution of model parameters given the observed data [90], [40].

2). Sparse Bayesian Learning (SBL)

SBL is widely applied in compressive sensing. The original motivation was to find a sparse representation for a target signal [107], [108]. Meanwhile, it has been applied by the system identification community to impose sparsity. Many identification problems are formulated to express an unknown function using basis functions from a pre-defined dictionary [68], [109]. Hence, a sparse representation can effectively avoid over-fitting and help detect network topology under the context of network inference. For example, SBL has been applied to inference of biochemical reaction networks [110] and online fault diagnosis for nonlinear power systems [111].

SBL adopts the Bayesian framework, where priors are introduced for model parameters. Nevertheless, instead of conducting a point estimate like MAP, SBL takes the mean of model parameters. As marginalizing the Bayesian model is intractable, SBL applies empirical Bayes to approximate the true distribution. This Bayesian relaxation is also applied in this thesis.

3). Kernel-Based identification and Gaussian process regression

The kernel method as a non-parametric method is widely applied in system identification community and its usage is still growing nowadays [112]–[117]. A thorough survey can be found in [118]. The kernel method works well for both nonlinear and linear models. It identifies the dynamics of a system rather than estimates model parameters. Generally speaking, the kernel method searches for a function that characterizes the dynamics of a system (e.g. impulse responses of a linear system) within a functional space of infinite dimension. As a functional space is associated to a unique kernel function, using a kernel function that is consistent with the dynamical property of the target system can greatly improve estimation accuracy.

The kernel method is intimately related to Gaussian process regression [74]. Hence, the identification problem raised by the kernel method can be recast under the Bayesian framework [119]. Consequently, Bayesian methods are applied to further improve the algorithm performance. One important instance is to use empirical Bayes to optimize the parameters of the kernel function that controls the property of the functions contained in the functional space [117]. Moreover, if these parameters also determine network topology, this Bayesian routine naturally accomplishes the task of topology detection [113].

4). Markov chain Monte Carlo method (MCMC)

MCMC is a tool to draw samples from a distribution. It is frequently used in system identification because, in many cases, there is no analytic solution to the integral of a Bayesian model. With MCMC, the integral can be well estimated from samples. Compared with SBL that approximates the true distribution analytically, MCMC generates samples that asymptotically distribute as the true one. Hence, many problems that are solved using deterministic approximations (e.g. empirical Bayes) can also be tackled by MCMC approaches. MCMC has been very active in identification of nonlinear dynamical systems [120]–[123]. While it was also applied to infer the topology of biological networks [124], [125], little attention has been paid to system dynamics.

One of the variants of MCMC is reversible jump MCMC (RJMCMC). RJMCMC draws samples from a distribution whose random variables have flexible dimensionality [126]. In this project, we apply RJMCMC to infer networks where the dimension of random variables is related to the unknown topology.

5). Prediction Error Minimization (PEM)

One of the most widely applied methods to identify linear systems is PEM. PEM is an indirect method to estimate a continuous-time (CT) model. It first identifies a discrete-time (DT) model from data and then converts it into a CT model. Although there exist methods to estimate a CT model directly using discrete-time data, they suffer from the difficulty in dealing with non-measurable time-derivatives [94]. PEM has a well-established property that, if the true system is contained in the proposed model structure and there are infinite data points, the estimated model will converge to the true system with variance dying out [89]. However, since the available data in practice are limited and noisy, the performance of PEM is degraded [91].

2.4 Conclusion

This chapter presents basic methodologies to infer biological networks. To infer a general network, a mathematical model is first postulated to describe the target network. With measured data of the network, the model is identified using system identification techniques. The estimated models reveal topology and internal dynamics of the unknown biological network. Moreover, systematic comparison of these models (using e.g. υ-gap [103], [127]) can help locate the target of treatment or find out dynamical changes due to mutations.

Various types of models can be used to describe a biological network on either microscopic (molecular population) or macroscopic levels (species concentration). A variety of methods have been developed to identify different types of dynamical systems. Usually, the goal of identification is to construct a model that captures dynamics (input-output relation) of the target system with small generalization errors for the purpose of, for example, control and prediction. Hence, two aspects are particularly important in modeling biological systems: model class and model parsimony. A rich model class is chosen to effectively describe system dynamics while model parsimony favors models with lower complexity to reduce generalization errors. Nevertheless, as the internal structure of a dynamical system is not a main concern, model structure is not part of identification in many cases.

Ignoring model structure is not a problem in some applications. For example, the input-

output relation of a linear system is invariant under linear transformation even if its internal dynamics completely change. However, to fully understand biological networks, it is imperative to learn about internal dynamics. As model structure is intimately related to internal dynamics, its detection must be embedded in the identification procedure. Hence, many prevalent identification methods are not suitable to infer biological networks. The main objective of this project is to develop novel methods for network inference. The main issues we concern include lack of prior knowledge, incomplete measurement of network species, requirement of sparse network topology, constraints on system stability, scaling computational cost to infer large-scale networks, etc.

This project applies black-box LTI systems and grey-box nonlinear systems to model biological networks. Identification of the proposed models is based on time series data. Little or even no prior knowledge of the target network is necessary for inference. Internal dynamics and topology of the target network are both inferred from data.

Chapter 3.

One-to-One method to infer dynamic networks

Compared to normal system identification problems, network inference includes an additional step: detecting network topology. Many statistical methods have been developed to infer network topology in a pairwise manner [14]–[16], [25]. For example, correlation-based methods estimate correlations between each pair of nodes and Winner-Granger causality measures dependence of each ordered pair of network units. These methods require no prior knowledge, scale well with the size of networks and are computationally cheap. They have been applied to infer large-scale networks using high-throughput data. However, the main disadvantage of these methods is that they are focused on learning connectivity of the target network whereas its internal dynamics are not fully explored.

This chapter discusses a method that, for some links in the network, finds both topology and dynamics. Moreover, just like the above methods, it scales. Furthermore, inference is purely based on time series data without any prior knowledge. We call this method one-to-one since it infers a link between each ordered pair of nodes.

This framework was initially proposed in [103] and further discussed in [102] and [128]. Later, this method was applied to infer the circadian clocks of *Arabidopsis* and Barley [103], [129]. Although the method is heuristic, as a co-author of some of the completed work (e.g. [102], [129]) and for the sake of further comparison with other proposed methods in this thesis, I would like to present more details about this method in this short chapter.

This chapter is organized as follows. Section 3.1 presents details of one-to-one. Section 3.2 discusses the strength and weakness of one-to-one. Finally, section 3.3 concludes the chapter and discusses the future research.

3.1 One-to-one method

We infer a network in a pairwise manner like correlation-based methods. The link between an ordered pair of nodes in a network is described by a SISO OE model as:

$$y(t) = G(q^{-1}, \theta)u(t) + e(t) + c \tag{3.1.1}$$

where $G(q^{-1})$ is a strictly proper transfer function:

(3.1.2)

$$G(q^{-1},\theta)=\frac{b_1q^{-n+1}+b_2q^{-n+2}+\cdots+b_n}{a_1q^{-n}+a_2q^{-n+1}+\cdots+1}$$

and q^{-1} is the time delay operator. $y(t)\in\mathbb{R}$ denotes the output of the system, $u(t)\in\mathbb{R}$ the input, $e(t)\in\mathbb{R}$ i.i.d. Gaussian noise and c a constant presenting the offset of the system. a and b are model parameters contained in θ and n is the system order. For an ordered pair of nodes, (x,y), we associate it with a directed link, $x\to y$ representing a regulation from x to y. Therefore, the corresponding OE model for this link takes x as the input and y as the output.

Model parameters θ, offset c and system order n are unknowns that must be determined. Hence, the aim of the inference problem is to identify OE models of each ordered pair of nodes from high-throughput time series data. For a network composed of N nodes, there are $N(N-1)$ possible links in total. Therefore, at least $N(N-1)$ OE models need to be identified. We use the prediction error minimization (PEM) method to identify these OE models (function tfest in Matlab).

The predictor of an OE model is [89]:

(3.1.3)

$$\hat{y}(t)=G(q^{-1},\theta)u(t)+c$$

where $\hat{y}(t)$ is the predicted output at time instance t given past measurements of outputs and inputs.

Model parsimony is essential for one-to-one because it avoids over-fitting and reduces sensitivity to undirected links that cause false positives. To impose model parsimony in accordance to high-throughput data, only 1st and 2nd order OE models are identified for each link. The AIC criterion is used to select the best system order. After modeling, we obtain a fully connected network, where each link is described by either a 1st or a 2nd order OE model. Model fitness measures how well estimated models reproduce the data. We will use this metric to evaluate confidence of inferred links. A threshold on fitness can be set to select true links. Consequently, the resulting network topology is reflected by the remaining links (in case of using thresholds) and their associated internal dynamics described by the OE models. In practice, to further reduce false positives, a stronger penalty for system order can be applied (e.g. AICc) or we can restrict the models to 1st order.

To conclude, the algorithm of one-to-one is stated below:

Algorithm One-to-one method (with thresholds)
1: Given high-throughput time series data of a network containing N nodes.
2: Set threshold T for model fitness.
2: ***For*** $i=1{:}N$ (index of nodes as inputs)
3: ***For*** $j=1{:}N$ (index of nodes as outputs)
4: ***If*** $i\neq j$ do
5: Estimate OE models of 1st and 2nd order: $\mathrm{model}=\mathrm{tfest}(\mathrm{data},\mathrm{order})$
6: Calculate model fitness: $\mathrm{fitness}=\mathrm{compare}(\mathrm{data},\mathrm{model})$
7: Calculate AIC: $\mathrm{AIC}=\mathrm{aic}(\mathrm{model})$

8: Keep the model with the lowest AIC
9: Store the model if its model fitness is higher than threshold T
10: ***end if***
11: ***End for***
12: ***End for***

Note that offset *c* in the model can be seen as the output of a zero order model with a step function as the input. Hence, the resulting OE model has two inputs: the actual input of the model and a second "virtual" input that represents offset *c*. The offset can be estimated along with transfer function $G(q^{-1}, \theta)$ using Matlab code $\mathrm{tfest}(\mathrm{data}, [\mathrm{order}, 0])$.

3.2 Discussion

One-to-one is straightforward to implement and has low computational cost. Therefore, it can be used to infer large-scale networks. In particular, one-to-one can be used to reveal interactions among nodes where a complete inference of the network is not needed. As one-to-one only estimates low order models, the method is suitable to deal with high-throughput data.

The methods using pairwise inference schemes (e.g. correlation-based methods and information theoretic methods) can suffer from three types of systematic prediction errors: fan-out error, fan-in error and cascade error [130]. Fan-out error happens when a node is controlling more than one other node. In this case, those regulated nodes usually show dependence in the inferred network even if they are not linked by directed edges in the true network. Fan-in error is caused by a node regulated by more than one other node. This is an intrinsic error of the pairwise inference framework. Cascade error is due to shortcuts between two nodes, where there is an indirect pathway from one node to the other.

One-to-one is robust to fan-out error because dependence among nodes is determined by the identified dynamics of regulations represented by models rather than by statistical tests of correlations. Nevertheless, one-to-one can still be influenced by fan-in and cascade errors. For fan-in error, each node in the network can only be controlled by at most one other node. One-to-one may still work with multiple inputs if they share similar dynamics.

Regarding cascade error, one-to-one cannot distinguish direct links from indirect ones thus leading to false positives. For example, if node A is controlled by node C via an intermediate node B and both links can be described by 1st models (Figure 4.2.1), this indirect link from node C to node A can be expressed by a 2nd model. One-to-one picks this indirect link with high fitness causing a false positive. This cascade error is due to the fact that regulations from other nodes to the target node are explored independently. In fact, given node B, node C has no contributions to node A. Unfortunately, one-to-one is not able to detect such dependence. Nevertheless, if an indirect link passes through many nodes or intermediate links have complex dynamics, this error may be avoided because an OE model up to 2nd order cannot represent complex dynamics of that indirect link. As a result, one-to-one filters out transitive pathways containing more than two direct links. Another way to reduce cascading error is to determine whether the product of the transfer functions from C to B and B to A is similar to the transfer function from C to A. If so, the inferred link from C to A is likely to be a false positive.

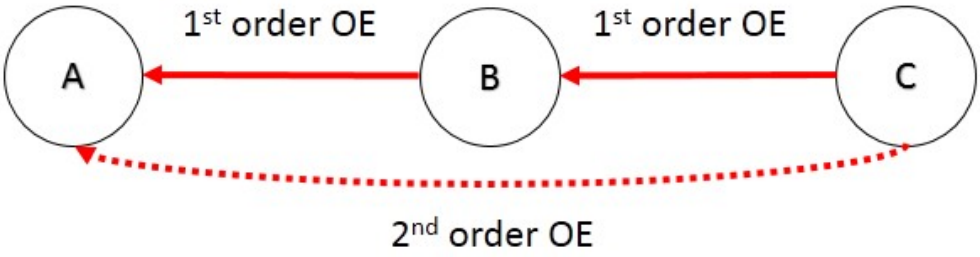


Figure 3.2.1: A chain link. A, B and C present three nodes. Red solid arrows denote direct links. The red dashed arrow presents a shortcut from C to A. Two direct links are described by 1st order OE models. Although the link from C to A does not exist, the relation between A and C can be well expressed by a 2nd order OE model if noise and offset are small.

3.3 Conclusion

This chapter presents a model-based method called one-to-one to infer biological networks using high-throughput data. Both network topology and internal dynamics of the target network are inferred from data. The network is reconstructed in a pairwise manner. An OE model is estimated for each ordered pair of nodes representing dynamics of a one-way regulation, where one node is treated as the input and the other as the output. Models are selected using the AIC criterion to avoid over-fitting. A threshold is set to discard models with low fitness.

One-to-one is easy to implement and can be applied to general large biological networks with no prior knowledge. This method demands very limited data because only low order models are estimated, thus suitable to deal with high-throughput experimental data with a small number of samples. The computational burden of one-to-one is light and scales quadratically with respect to the number of nodes. Therefore, it can be used to infer large-scale networks.

Nevertheless, one-to-one has several drawbacks. First of all, it is only suitable to infer networks consisting of chain links. Systematic errors occur if a node is controlled by more than one node. One-to-one also suffers from cascade error if fast dynamics exist in a pathway that can be well approximated by low order systems. Hence, this method cannot distinguish direct links from indirect links, leading to false positives. As the complexity of the target network grows, the performance of one-to-one is expected to degrade since internal dynamics of the network cannot be fully interpreted by OE models up to 2nd order. Additionally, inputs of the network are not considered by one-to-one.

For future developments, one-to-one can be improved in two ways. First, more advanced mathematical models should be used to describe networks to capture complex internal dynamics. Second, the proposed inference method should infer interactions among nodes simultaneously rather than independently to avoid cascade error and fan-in error.

Chapter 4.

Applied Sparse Bayesian Learning to sparse network inference

Many large biological networks, e.g. metabolic and genetic networks have sparse network topologies [131]–[133]. Indeed, most molecules have great affinity and can bind to only a small number of other molecules [19]. Hence, within the field of molecular biology, it is reasonable to impose sparse network topologies during inference. Different methods handle sparsity constraints in different ways. For example, mutual information based methods measure dependence between each pair of nodes, which is used to decide the network topology (i.e. selection of edges) based on a heuristic threshold [134], [135]. Several model-based methods, that identify internal dynamics of a biological system, introduce penalties or prior distributions to implicitly or explicitly impose sparse network topologies during inference. Examples include Gaussian process via estimation of hyperparameters of kernel functions [136], [137], compressive sensing via Lasso type penalties [138] and hierarchical Bayesian regression via prior distribution of model parameters [139]. Network topology and internal dynamics are intimately related and they must be learned simultaneously. Incorporating sparsity as prior knowledge of network topology into inference can greatly improve accuracy of estimated models.

Although the one-to-one method presented in the previous chapter is a model-based approach, it produces sparse networks through a heuristic threshold for model fitness. One-to-one infers a network in a pairwise manner that highly relies on data-fitting. Model parsimony is only considered locally for individual edges (i.e. model selection via AIC for estimated 1^{st} and 2^{nd} order models) whereas promoting sparse network topologies requires global model parsimony for the entire network, i.e. models for some edges are directly eliminated during identification to reduce global model complexity (e.g. sum of equally weighted system order of all edges). Although inferring all edges simultaneously is essential to avoid systematic errors of inference, an exact optimization of model parsimony can be computationally prohibitive for large-scale networks because it demands an exhaustive search of all possible topologies.

This chapter resorts to compressive sensing frameworks to relax the problem of model

selection. A weighted penalty is used to impose sparsity of network topology. While the weight for penalties normally requires tuning (e.g. cross validation), we apply a technique called sparse Bayesian learning (SBL) that is tuning-free.

Linear multivariable ARX models are used to describe sparse biological networks. Identification of the model is performed directly from time series data without any prior knowledge. The novelty of the proposed method lies on the combination of element and group sparse Bayesian learning to introduce penalties for complexity, both in terms of element (order of non-zero connections) and group sparsity (network topology). The framework is further extended to nonlinear ARX models. Data from simulated random networks indicate that our method, on average, performs better than previous methods, with considerably higher performance when networks have few links (as, for example, in the case of as ring structures). Simulation results will be presented in chapter 6 to have a comparison with other proposed methods.

This chapter is organized as follows. Section 4.1 introduces the background of sparse network inference. Section 4.2 discusses commonly used MAP approaches and Bayesian frameworks to impose sparsity patterns. Section 4.3 introduces full Bayesian models and Bayesian techniques to tackle intractable Bayesian estimation. Section 4.4 presents variational representation of sparsity inducing prior distributions and section 4.5 discusses SBL and its underlying mechanisms. Section 4.6 introduces multivariable ARX models, while section 4.7 formulates the network reconstruction problem. Section 4.8 proposes combined group and element sparse Bayesian learning (GESBL) and section 4.9 presents algorithms to solve the proposed inference problem. Section 4.10 extends the framework to nonlinear ARX models. Section 4.11 concludes the chapter.

4.1 Sparse network inference

Sparsity is an inherent property of many practical networks. In biology, most molecules bind with a small number of other molecules. Hence, sparsity can be used as a constraint to model networks and compensate for the sometimes low number of samples and high amount of noise.

As we assume no prior knowledge of the target network and would rather infer the network purely based on measured data, black-box linear models defined in terms of inputs and outputs can be adopted to describe networks, including ARX, ARMAX and Box-Jenkins. Standard system identification methods, such as the prediction error method (PEM) or Maximum-likelihood (ML), are applicable to identify these models [89], [140], [141]. However, these methods alone do not capture topology and sparsity with finite source of data. For example, assuming no prior knowledge of the topology, PEM generates full transfer matrices even if the ground truths are sparse [113]. That is because these methods only identify input-output relationships of a system via data-fitting rather than exploring internal structure and dynamics.

A remedy is to promote model parsimony to favor sparsity. Under the Bayesian framework, model parsimony is realized by introducing prior distributions. Bayesian inference methods such as hierarchical Bayesian regression (HBR) [90] are effective to infer sparse networks. Although these methods enable evaluation of uncertainty of inferred networks, they require

an exhaustive search of all possible topologies, thus being computationally demanding. Maximum a posteriori methods (Type I method) including least absolute shrinkage and selection operator (LASSO), Tikhonov regularization, FOcal Underdetermined System Solver (FOCUSS) and SGL are all methods that penalize model complexity through deterministic optimization [107], [142]–[144]. For example, a LASSO algorithm was used to infer the topology of a linear MIMO system from steady-state data [145]. Similar work inferred sparse multivariable ARX models with the fixed polynomial order using a greedy algorithm, Block Orthogonal Matching Pursuit (BOMP) [146], to favor sparse network topologies [147].

Whilst these approaches effectively reduce over-fitting, the weighting variable, which controls the trade-off between data-fitting and model complexity (sparsity), must be *a priori* chosen or evaluated independently, using methods such as cross-validation. Unfortunately, this increases the computational burden, causes information waste and can bias the solution.

There are, however, alternative methods that do not require tuning variables. One such method is sparse Bayesian learning (Type II method). SBL is a well-known method in machine learning and widely applied for compressive sensing. SBL can be recast as a specific type of MAP methods, where the particularity lies on the usage of inseparable penalties to promote sparsity [71], [108], [148], [149]. SBL has also been introduced to system identification community. It was applied to identify grey-box nonlinear systems [68], [110]. The nonlinear model structure is captured either by element SBL or by group SBL, depending on the type of data available and selected model classes.

Obtaining sparse representations, as in modeling biochemical systems, requires both element and group sparsity. However, currently there are no methods that combine these two types of sparsity constraints. Hence, this chapter considers the parametric identification of a sparse network described by a multivariable ARX model. The goal is two-fold: (a) to infer the network topology and (b) to achieve accurate estimation of model parameters including the polynomial order.

4.2 Bayesian approaches and point estimation via optimization

Many system identification problems can be formulated and solved as a linear regression problem, depending on the structure of the postulated models. Various techniques have been developed to solve regression problems whose parameter vectors have special properties such as sparsity. Under the context of system identification, attributes of the formulated parameter vector are related to the model class that is chosen in accordance to specific applications. We consider methods that are developed to solve sparse linear regression problems.

A linear regression model is defined as follows:

$$y = \Phi w + \varepsilon \tag{4.2.1}$$

where $\Phi \in \mathbb{R}^{n\times m}$ is a dictionary matrix consisting of basis vectors, $w \in \mathbb{R}^m$ is a sparse vector (many elements are 0) of unknown parameters, $y \in \mathbb{R}^n$ is observed data and $\varepsilon \in \mathbb{R}^n$ is Gaussian noise with an isotropic covariance matrix [90]. The objective is to estimate parameter vector w, given the observed data and dictionary matrix. In many practical cases, a large number of basis vectors are present relative to the number of data points ($n \ll m$). Therefore, estimating w is an under-determined problem. Without extra constraints on w,

there may exist infinite solutions.

4.2.1. Point estimation via deterministic optimization

A fundamental method to solve a linear regression problem is least squares. The spirit of this method is to find w that can best fit the observed data in the sense of minimizing fitting errors measured by sum of squares [150]:

(4.2.2)

$$\widehat{w} = \arg\min_{w} \|y - \Phi w\|_2^2$$

The solution set is equivalent to the one of a linear equation: $\Phi'\Phi w = y$. Since Φ is fat and not full column rank, the solution is not unique and forms an affine set. More importantly, the solution is not sparse due to noisy measurements.

A typical remedy is to add a penalty (regularizer) to the data-fitting term leading to a canonical regularized optimization problem [108]:

(4.2.3)

$$w = \arg\min_{w} \|y - \Phi w\|_2^2 + \lambda g(w)$$

where the least squares term measures the data-fitting error, $g(\cdot)$ is a fixed function to penalize vector w and λ is a non-negative tuning variable that controls the trade-off between data-fitting and regularization. Evaluation of λ cannot be directly embedded into optimization as it may cause an ill-posed problem. This variable is normally estimated using, for example, cross validation to reduce generalization errors.

The harshest penalty for sparsity is ℓ_0 norm that penalizes element-wise deviation from zero, where $\|w\|_0$ equates to the number of non-zero elements in w. Nevertheless, finding the global minimum of the resulting optimization problem is combinatorial thus NP-hard [108]. A remedy is to apply other penalties that are relaxations of ℓ_0 norm. Usually, $g(\cdot)$ is selected to be permutation invariant (i.e. $g(w) = g(Pw)$ where P is a permutation matrix), sign invariant (i.e. $g(w) = g(|w|)$) and concave on the positive orthant [151]. Generally, the solution is sparser if the penalty is more concave, thus better approximating ℓ_0 norm. However, the trade-off is that the number of local optima increases combinatorically as the concavity of penalties [107]. By using such penalties, sparsity of suboptimal solutions is guaranteed by Theorem 4.2.1.

Theorem 4.2.1: Every local minimum of the cost function, $f(w) = \|y - \Phi w\|_2^2 + \lambda g(w)$ has at most n non-zero elements regardless of the tuning variable λ if $g(\cdot)$ is permutation invariant, sign invariant and concave on the positive orthant.

Proof:

Assume w^* is a local minimum of the cost function, $f(w)$ we have that:

$$\exists \epsilon > 0 \text{ such that } \forall w \in C_1 = \{w | \ \|w - w^*\|_2 < \epsilon\}$$
$$f(w) > f(w^*)$$

Let $e^* = y - \Phi w^*$ and $C_2 = \{w | \ y - \Phi w = e^*\}$. For $w \in C_1 \cap C_2$, we have that:

$$g(w) > g(w^*)$$

So, w^* is also a local minimum of the following optimization problem:

$$\arg\min_{w} g(w)$$

Subject to: $\Phi w = y - e^*$

Based on the theorem in [151], every local minimum of the above optimization problem has at most n non-zero elements.

Theorem 4.2.1 implies that even if the global optimum cannot be achieved, suboptimal solutions of (4.2.3) are always sparse.

We illustrate some widely applied regularizers that all satisfy the conditions in Theorem 4.2.1. One well known algorithm is LASSO that uses ℓ_1-norm as the penalty [150], [144]:

(4.2.4)

$$w = \arg\min_{w} \|y - \Phi w\|_2^2 + \lambda \sum_{r=1}^{m} |w_r|$$

The penalty function $g(w) = \sum_{r=1}^{m} |w_r|$ is the best convex relaxation to ℓ_0 norm. Actually, on the positive orthant, it is a linear function. Since the cost function is convex, the solutions are all global optima.

Another proposed regularized optimization problem adopts a penalty in the form of Gaussian entropy [107]:

(4.2.5)

$$w = \arg\min_{w} \|y - \Phi w\|_2^2 + \lambda \sum_{r=1}^{m} \log |w_r + \epsilon|$$

where $\epsilon > 0$.

As ϵ approaches 0, the penalty becomes steep around the origin thus resulting in an extremely sparse solution.

The last one generalizes the LASSO type algorithm and is called FOCUSS [152], [153]:

(4.2.6)

$$w = \arg\min_{w} \|y - \Phi w\|_2^2 + \lambda \sum_{r=1}^{m} |w_r|^p$$

where $p \in [0,1]$.

The penalty in this case is l_p-norm (not a valid norm defined in a strict sense). As p approaches 0, the penalty becomes strongly concave and leads to sparse solutions. Nevertheless, the number of suboptimal solutions also increases combinatorially.

The algorithms presented above are designed to solve regression problems whose parameter vectors are element sparse. Certainly, parameter vectors are allowed to have other types of sparsity, for example, group sparsity and combination of both element and group sparsity. Penalties have also been developed to impose these kinds of sparsity.

If a parameter vector is group sparse, group LASSO (GLASSO) as a variant of LASSO can be used [154]–[156]:

(4.2.7)

$$w = \arg\min_{w} \|y - \Phi w\|_2^2 + \lambda \sum_{r=1}^{k} \|w_r\|$$

where $\|\cdot\|$ denotes the Euclidean norm and w_r is the rth group of w. This regularizer penalizes the size of each group of the parameter vector.

A direct blend of GLASSO and LASSO called sparse group LASSO (SGL) was developed to estimate element and group sparse parameter vectors [143], [157]:

(4.2.8)

$$w = \arg\min_{w} \|y - \Phi w\|_2^2 + \lambda_1 \sum_{r=1}^{m} |w_r| + \lambda_2 \sum_{r=1}^{k} \|w_r\|$$

It should be noticed that all the algorithms above assign the same form of penalty functions to each element and group of w, and these elements and groups are penalized independently. We will introduce inseparable penalties induced in the Bayesian framework which not only impose sparse solutions but also tremendously reduce the number of local optima.

4.2.2 Bayesian framework of linear regression

Designing penalties to regularize parameter vectors directly is non-trivial and, in most circumstances, intuitive. For example, one may choose Tikhonov regularization [158] to avoid over-fitting but ℓ_1 norm to impose element sparsity [159], [145]. Furthermore, evaluation of the tuning variable that controls the trade-off between data-fitting and regularization often demands extra computation efforts. In contrast, many problems that are difficult to formulate directly in the way of deterministic optimization can be easily cast under the Bayesian framework. Formulation of identification problems is naturally derived from the stochastic properties of a system. Unknown parameters or functions to be determined are treated as random quantities and used to formulate full Bayesian models. With Bayesian techniques, prior distributions can be updated during inference using observed data to counteract possible bias caused by prior knowledge. More importantly, Bayesian frameworks enable evaluation of model uncertainty in contrast to a point estimation by optimization, which is essential for other applications (e.g. simulation and prediction).

The linear regression model (4.2.1) raises a conditional distribution of the observed data given the parameter vector. This distribution is normally called likelihood function:

(4.2.9)

$$p(y|w) \sim \mathcal{N}(w|\Phi w, \lambda)$$

where λ is the variance of the Gaussian noise.

By maximizing the likelihood, $\max_{w} p(y|w)$, a point estimate of w is found from the parameter space so that the resulting model is most likely to reproduce the observed data. It is straightforward to see that the resulting optimization problem is the same with least squares.

In system identification, such an approach is called maximum likelihood (ML). The main advantage of ML is estimation consistency. With infinite source of data, the estimate will converge to the ground truth [89], [160]. Nevertheless, due to the lack of data in practice, the solution may not be sparse and unique.

Under the Bayesian framework, we incorporate prior knowledge to assist inference. According to Bayesian paradigms, model parameters are treated as random variables. A prior distribution for w is introduced to reflect initial belief. The knowledge of w is further improved once the observed data are available. Consequently, the uncertainty of model parameters is evaluated from the posterior distribution based on Bayes' rules:

(4.2.10)

$$p(w|y) = \frac{p(y|w)p(w)}{p(y)}$$

where

$$p(y) = \int p(y|w)p(w)dw \tag{4.2.11}$$

By estimating w as the maximum of the posterior distribution: $\max_w p(y|w)p(w)$, we achieve another point estimation. This approach is also known as maximum a posteriori (MAP) [40]. It has a direct link with regularized optimization methods. By introducing a prior in the form of $p(w) \propto \exp\left[-\frac{1}{2}g(w)\right]$, MAP results in the same regularized optimization problem (4.2.3). Penalties that are used to enforce sparse solutions can be introduced as factorial priors (permutation invariant) for the parameter vector in Bayesian models, i.e. elements of w are independent random variables. Therefore, LASSO is equivalent to introducing a Laplace prior ($p(w) \propto exp\left[-\frac{1}{2}\sum_{r=1}^{m}|w_r|\right]$), Gaussian entropy to using a Jeffrey prior ($p(w) \propto \prod_{r=1}^{m}\frac{1}{|w_r+\epsilon|}$, $\epsilon > 0$) and FOCUSS to applying a generalized Gaussian prior ($p(w) \propto exp\left[-\frac{1}{2}\sum_{r=1}^{m}|w_r|^p\right]$, $p \in [0,1]$). These priors are all sparse inducing priors widely used in Bayesian estimation [161], [107].

Bayesian inference prefers to estimate w by evaluating the entire parameter space characterized by a probabilistic measure (e.g. mean of posterior distribution), which requires a complete knowledge of the Bayesian model. However, the posterior distribution, $p(w|y)$ is normally known up to a constant since the integral (4.2.11) is usually intractable. This integral is inevitable if we want to evaluate the confidence of w within a range (i.e. $P(w_1 < w \leq w_2) \propto \int_{w_1}^{w_2} p(y|w)p(w)dw$). Additionally, for the purpose of prediction, we would like to calculate $p(y|y_o) = \int p(y|w)p(w|y_o)dw$ where y_o is the observed output. As a result, approximations must be made to the true Bayesian model as a compromise, namely approximate inference.

Another remarkable feature of the Bayesian framework is that it allows to update prior distributions during inference. Prior distributions induced by initial belief of a system can be inaccurate, thus causing biased estimation. Adjusting prior distributions during inference forms a feedback loop in the inference procedure, which generates a more robust estimate.

4.3 Full Bayes and Empirical Bayes approaches

To construct priors whose properties can be adjusted with a sufficient degree of freedom, one can parametrize prior distributions by additional variables called hyperparameters. Full Bayesian treatment also regards hyperparameters as random variables and introduces priors for them (hyperpriors) [40], [90]. For example, a Gaussian prior is conditioned on the scale parameter and an Inverse-Gamma distribution can be used as the hyperprior. As a result, the

final probabilistic model has a hierarchical structure. The number of layers can be increased by introducing other hyperparameters, each of which is conditionally independent on the former one [106]. Figure 4.3.1 shows a two-layer hierarchical model compared with a three-layer counterpart. A full Bayesian model is described by a joint posterior distribution of model parameters and hyperparameters. For a two-layer model, we have:

(4.3.1)

$$p(w,\beta|y) = \frac{p(y|w)p(w|\beta)p(\beta)}{p(y)}$$

There are other ways to construct flexible prior distributions. We will discuss more details in the next section. For the sake of simplicity, we focus on full Bayesian models with a hierarchical structure in this section.

In practice, we evaluate the uncertainty of model parameters, w, which is indicated by the marginal posterior distribution:

(4.3.2)

$$p(w|y) = \frac{p(y|w)p(w)}{p(y)} = \int p(w,\beta|y)d\beta$$

where

(4.3.3)

$$p(w) = \int p(w|\beta)p(\beta)dw$$

The construction of such hierarchical models seems cumbersome. As the actual prior is $p(w) = \int p(w|\beta)p(\beta)dw$ instead of $p(w|\beta)$, the question is why not use $p(w)$ directly?

The reason to apply hierarchical models is three-fold. First, we should note that although the expression of $p(w)$ may be complex or even intractable, its decomposition $p(w|\beta)$ and $p(\beta)$ are analytical. In other words, hierarchical structure provides a flexible way to postulate novel priors. Another benefit is that $p(w,\beta|y)$ can be easily evaluated up to a normalization constant even if $p(w|y)$ does not have a closed form. In the end, we have access to $p(w|y)$ by exploring $p(w,\beta|y)$ numerically using, for example, sampling schemes [162], [40]. Second, introducing hyperparameters enriches ways to manipulate complex Bayesian models. For example, empirical Bayes is one of the powerful Bayesian techniques that can be used to approximate intractable Bayesian models where hyperparameters are optimized to minimize the approximation error [90]. Finally, introducing hyperparameters allows us to solve complex inference problems more efficiently using algorithms such as expectation-maximization [163].

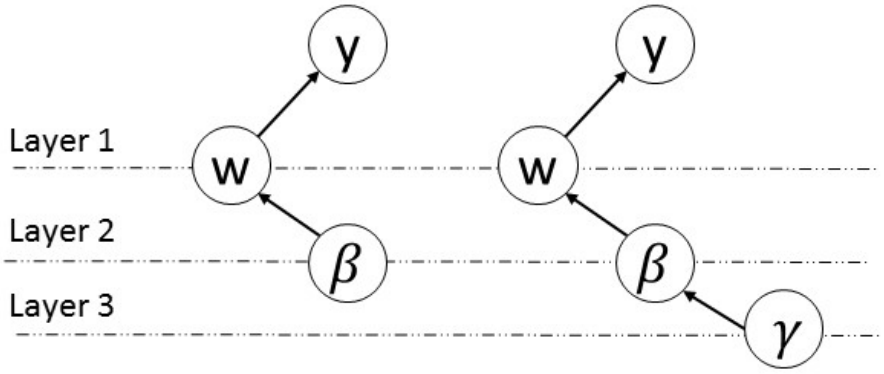


Figure 4.3.1: Hierarchical Bayesian model. The left one has two layers and the right one has three layers. Nodes present random variables and arrows indicate dependence.

Due to the intractability of integral (4.3.2) or (4.3.3), $p(w|y)$ is usually not analytical. To deal with this problem, approximation schemes are applied to the true distribution, $p(w|y)$. These methods fall broadly into two categories: stochastic and deterministic approximations [90]. Stochastic techniques rely on numerical sampling methods (e.g. Markov chain Monte Carlo [164]) that are able to draw samples from the full Bayesian model, $p(w,\beta|y)$. Under some mild conditions and given infinite computational resource, the generated samples are distributed asymptotically as the true Bayesian model, $p(w,\beta|y)$ [90]. By using these samples, one can establish the empirical distribution, $\hat{p}(w|y)$ that converges to the true marginal distribution, $p(w|y)$ as the number of samples grows [162], [165]. In practice, sampling methods are often computationally demanding, thus only suitable for small scale problems. We will discuss stochastic approximations further in chapter 6.

In contrast to stochastic techniques, deterministic approximation schemes scale well to large applications. These methods approximate the true posterior distribution analytically [90]. Deterministic approximations consist of two main steps. A candidate distribution, $\hat{p}(w|y)$ is first introduced to approximate $p(w|y)$. Then, a criterion that measures the distance between the approximate and the true distribution ($d(\hat{p}(w|y), p(w|y) \geq 0$ where $d(\cdot,\cdot)$ can be a metric in some applications) is designed. Finally, the candidate distribution, $\hat{p}(w|y)$ is optimized to minimize the approximation error that is measured by the metric. Two widely applied deterministic approximations are empirical Bayes and variational inference. In general, $\hat{p}(w|y)$ in empirical Bayes is constructed using prior, $p(w|\beta)$, that is conditioned on hyperparameters. The hyperparameters are optimized to minimize the gap between $\hat{p}(w|y)$ and $p(w|y)$. In contrast, variational inference expresses $\hat{p}(w,\beta|y) = q(w|y)q(\beta|y)$ as the factorial of mean-filed, within which factors are optimized instead of hyperparameters. Consequently, the true $p(w|y)$ is approximated by $q(w|y)$. We will focus on empirical Bayes in this thesis as its formulation is more straightforward and its underlying mechanisms are easier to analyze.

To be detailed, empirical Bayes can use part of the full Bayesian model to construct approximate distributions. The prior, $p(w)$ is expressed in a hierarchical form (e.g. $p(w) = \int p(w|\beta)p(\beta)dw$). Normally, the bottom layer of the hierarchical model is discarded and the corresponding hyperparameter is no longer a random variable but an unknown deterministic parameter to be estimated.

Figure 4.3.2 shows a two-layer hierarchical model compared with the one used for approximations by empirical Bayes. Since the hyperparameter is not a random variable, the empirical Bayes model only has one layer. Hyperparameter, β becomes a tuning variable of the prior expressed as $p(w|\beta)$.

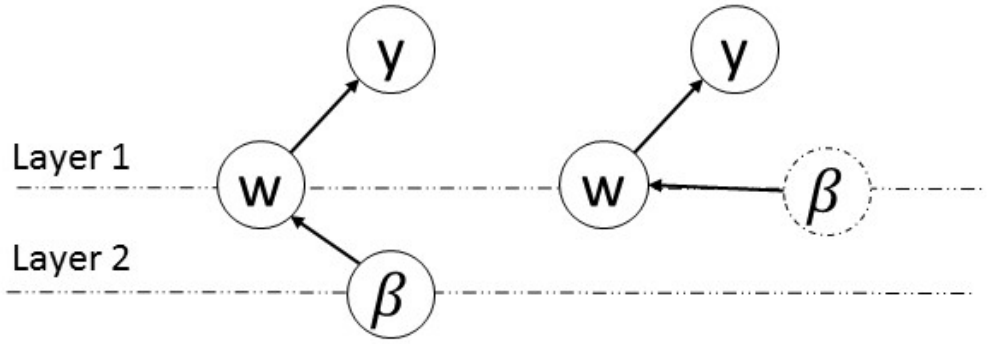


Figure 4.3.2: Hierarchical Bayesian model and empirical Bayes model. The left one (hierarchical Bayesian model) has two layers and the right one (empirical Bayes model) has only one layer. Nodes in solid circles present random variables whilst the ones in dashed circle are deterministic but unknown parameters. Edges indicate independence.

Empirical Bayes approximates the true posterior distribution, $p(w|y)$ by the one conditioned on the fixed but unknown hyperparameter, β:

(4.3.4)

$$p(w|y) \approx p(w|y,\beta) = \frac{p(y|w)p(w|\beta)}{p(y|\beta)}$$

Normally, $p(w|\beta)$ is carefully chosen so that $p(w|y,\beta)$ has an analytical expression. A traditional way is to introduce a conjugate prior for w given likelihood $p(y|w)$ [40].

The hyperparameter, β is estimated by the maximum likelihood method:

(4.3.5)

$$\hat{\beta} = \arg\max_{\beta} p(y|\beta)$$

where random variable w is marginalized out from the joint distribution, $p(y|\beta) = \int p(y|w)p(w|\beta)dw$. Estimating hyperparameters in this way is called evidence maximization or type II maximization [90].

An alternative way to interpret empirical Bayes is to treat $p(w|\beta)$ as the prior without introducing hyperpriors, where the density function is parametrized by hyperparameter β and β is deterministic but unknown. As hyperparameter β controls the property of $p(w|\beta)$ (e.g. length scale of distributions), empirical Bayes actually optimizes the prior, $p(w|\beta)$ through β using (4.3.5) so that the prior better reflects the nature of random variable w according to observed data. With this optimal prior distribution $p(w|\hat{\beta})$, the posterior distribution of w is:

(4.3.6)

$$p(w|y,\hat{\beta}) = \frac{p(y|w)p(w|\hat{\beta})}{p(y)}$$

Although introducing hyperparameters via the hierarchical structure of priors is straightforward, the criterion of empirical Bayes formulated in this way to measure the distance between $\hat{p}(w|y)$ and $p(w|y)$ is implicit. However, by applying an alternative representation of priors, we will have a clearer view and better insight of the underlying mechanisms of empirical Bayes.

4.4 Variational representation

In Bayesian probabilistic models, Gaussian priors are widely used because, given a logarithmically quadratic likelihood, the integrals of the resulting posterior distributions that are normally required by, for example, model prediction and parameter estimation have analytical solutions. Nevertheless, in many cases, Gaussian priors are not consistent with prior knowledge thus causing biases to inference results. For example, a sparse prior with heavy tails that are not exponentially bounded can reflect sparse properties of parameters whereas a Gaussian prior cannot. In practice, a much richer class of priors is required. Unfortunately, these non-Gaussian distributions often cause intractable problems. Variational representation of probability distributions is an alternative way to accommodate non-gaussianity so as to take advantage of tractability of Gaussian models [166]. This kind of representation allows to describe non-Gaussian densities in a deterministic way using Gaussian distributions. Hence, the merit of Gaussian models is retrieved when applying deterministic approximation techniques. Applications of variational representation include variational inference and sparse Bayesian learning.

There are mainly three types of variational representations: a) convex bounding, b) hyperpriors and c) variational Bayes [166]. In particular, convex bounding and hyperpriors are mainly used in empirical Bayes while variational Bayes is applied in variational inference.

Convex bounding (convex type representation) is based on a theorem of convex analysis that almost all the convex functions can be presented by the supremum of affine functions as $f(w) = \sup_{\beta} \beta w - f^*(\beta)$ where $f(\cdot)$ is a convex function and $f^*(\cdot)$ is the convex conjugate of $f(\cdot)$ [150]. The target density function is first transformed into a convex function (e.g. $f(w) = \log(p(w))$) and then bounded by a set of affine functions [167]. In the end, β turns out to be the hyperparameter of the approximate density function.

Hyperpriors (integral type representation) introduces latent random variables (hyperparameters) along with their priors (hyperpriors). The target density is described in a hierarchical structure, where latent variables are integrated out from the joint density (i.e. $p(w) = \int p(w|\beta)p(\beta)d\beta$). Hyperparameter β can be the variance or scale parameter of the conditional distribution, $p(w|\beta)$ [71]. Obviously, the priors introduced in the last section belong to this category.

Instead of characterizing densities via hyperparameters, variational Bayes expresses a distribution in the form of factorial mean-field (e.g. $\hat{p}(w) = \prod_{i=1}^{n} p_i(w_i)$). Variational inference approximates densities based on the theory of calculus of variations. Factors,

$p_i(w_i)$ of the mean-field are estimated to minimize the distance (e.g. measured by Kullback-Leibler divergence) between the approximate and the target density. Consequently, $p(w_1, w_2, \dots, w_n) \approx \prod_{i=1}^{n} p_i(w_i)$ [90].

4.4.1 Convex type variational representation

In this type of representation, densities are described by the supremum over Gaussian densities [166]:

(4.4.1)

$$p(w) = \sup_{\beta>0} \mathcal{N}(w|0, \beta^{-1})\varphi(\beta)$$

where $\varphi(\beta)$ is a positive function.

The expression indicates that a sufficient condition for a density to have a convex type representation is that $-\log p(\sqrt{w})$ is closed and concave on $(0, \infty)$. The following theorem also reveals the necessary condition for (4.4.1) to hold.

Theorem 4.4.1 [166], [107]**:** A probability density $p(w) = \exp[-g(w^2)]$ can be expressed in a convex type variational form if and only if $-\log p\left(\sqrt{w}\right) = g(w)$ is concave on $(0, \infty)$ where $g^*(\cdot)$ is a concave conjugate of $g(\cdot)$ and

$$\varphi(\beta) = \sqrt{\frac{2\pi}{\beta}} \exp\left[g^*\left(\frac{\beta}{2}\right)\right]$$

Many widely known densities can be expressed in this variational form including Generalized Gaussian containing Laplace distribution ($exp\ (-\gamma|w|^p)$ with $p \leq 2$), Student's t and symmetric α-stable densities with $0 < \alpha \leq 2$. All these densities can be used as sparse inducing priors. This property of variational representation motivates a specific classification of distributions.

Definition 4.4.1 [166]**:** A symmetric probability density, $p(w)$ is strongly super-gaussian if $-\log p\left(\sqrt{w}\right) = g(w)$ is concave on $(0, \infty)$ and strongly sub-gaussian if $-\log p\left(\sqrt{w}\right) = g(w)$ is convex on $(0, \infty)$.

Almost all the sparse inducing priors are super-gaussian [108]. These priors have heavy tails that lead to a more concentrated peak around 0 compared to a normal Gaussian distribution. In practical applications where collected data are noisy, sparse inducing priors effectively accommodate outliers while a Gaussian distribution shifts away from zero mean, thus failing to impose sparse results.

4.4.2 Integral type variational representation

In contrast to convex type representation, integral type representation describes a density by an integral over the scale parameter of a Gaussian distribution [163]:

(4.4.2)

$$p(w) = \int_0^{\infty} \mathcal{N}(w|0, \beta) d\mu(\beta)$$

where $\mu(\cdot)$ is non-decreasing and bounded on $(0,\infty)$.

A choice for $\mu(\cdot)$ is a cumulative distribution function, $F(\beta)$. Hence, the variational representation becomes:

(4.4.3)

$$p(w)=\int_0^\infty \mathcal{N}(w|0,\beta)p(\beta)d\beta$$

where $p(\beta)$ is the density function corresponding to $F(\beta)$.

The density, $p(w)$ is expressed in a hierarchical form with a latent random variable to be the scale parameter of a Gaussian distribution. For example, if the hyperprior, $p(\beta)$ is exponentially distributed, the resulting $p(w)$ is a Laplace distribution [168]. The following theorem states the necessary and sufficient condition for integral type variational representation.

Theorem 4.4.2 [166]**:** A density, $p(w)$ can be expressed in the integral type variational form if and only if $p\left(\sqrt{w}\right)$ is completely monotonic on $(0,\infty)$.

It should be noticed that functions that are represented in the integral type variational form can also be described by convex type representation [166]. Hence, convex type representation is more general than the integral type. In what follows, we will use convex type representation to develop novel inference methods.

4.5 Sparse Bayesian Learning

Sparse Bayesian learning (SBL) is originally developed to solve linear regression problems whose parameter vector is sparse using the technique of empirical Bayes [71].

From the Bayesian perspective, we can establish the posterior distribution for the model, $p(w|y)$ using the likelihood function, $p(y|w)$ and an introduced factorial prior distribution, $p(w)$ that is able to induce sparsity (sparse inducing priors):

(4.5.1)

$$p(y|w)\sim\mathcal{N}(y|\Phi w,\lambda)$$
$$p(w)\propto\exp\left[-\frac{1}{2}g(w)\right]$$
$$p(w|y)\propto p(y|w)p(w)$$

where λ is the variance of the Gaussian noise.

With a sparse inducing prior, the posterior distribution is normally skewed and may have multiple modes. Instead of searching for the most likely value of w, SBL takes the posterior mean as the estimate. Since the integral of expectation is intractable, SBL applies empirical Bayes to approximate the true posterior distribution.

SBL uses convex type variational representation to describe the sparse inducing prior, $p(w)$:

(4.5.2)

$$p(w)=\prod_{r=1}^{m}p(w_r)=\max_{\beta\geq 0}\mathcal{N}(w|\mathbf{0},B)\varphi^{\beta}(\beta)$$

$$p(w_r) = \max_{\beta_r \geq 0} \mathcal{N}(w_r|\mathbf{0}, B_r)\, \varphi_r^{\beta}(\beta_r)$$

Based on the variational representation, we get a lower bound of the prior that is parametrized by hyperparameter β:

(4.5.3)

$$p(w) \geq \hat{p}(w|\beta) = \mathcal{N}(w|\mathbf{0}, B)\varphi^{\beta}(\beta)$$

The lower bound, $\hat{p}(w|\beta)$ is used to approximate the true prior, $p(w)$. Although the lower bound density, $\hat{p}(w|\beta)$ is improper (i.e. its integral is not 1), we can still achieve a normalized posterior distribution of w. As the mean of the Gaussian likelihood is a linear function of w, $\hat{p}(w|\beta)$ is the conjugate prior thus resulting in a Gaussian posterior distribution:

(4.5.4)

$$\hat{p}(w|y,\beta) = \frac{p(y|w)\hat{p}(w|\beta)}{\int p(y|w)\hat{p}(w|\beta)dw} = \mathcal{N}(w|\mu_\beta, \Sigma_\beta)$$

where $\mathrm{B} = diag\{\beta\}$ and

$$\Sigma_\beta = \mathrm{B} - \mathrm{B}\Phi'(\lambda I + \Phi\mathrm{B}\Phi')^{-1}\Phi$$
$$\mu_\beta = \mathrm{B}\Phi'(\lambda I + \Phi\mathrm{B}\Phi')^{-1}y$$

SBL uses $\hat{p}(w|y,\beta)$ to approximate the true $p(w|y)$ so that the posterior mean can be easily calculated due to gaussianity. The gap between $p(w|y)$ and $\hat{p}(w|y)$ is measured as $d(p(w),\hat{p}(w)) = \int p(y|w)|p(w) - \hat{p}(w)|dw$ that is the misaligned mass between the true prior, $p(w)$ and the approximate, $\hat{p}(w)$ weighted by the likelihood, $p(y|w)$ [108]. It is easy to prove that function $d(\cdot,\cdot)$ is a valid metric on the convex cone consisting of non-negative functions on $\mathbb{R}^m$. Hyperpameter β is optimized to minimize the metric:

(4.5.5)

$$\begin{aligned}
\hat{\beta} &= \arg\min_{\beta} \int p(y|w)|p(w) - \hat{p}(w)|dw \\
&= \arg\min_{\beta} -\int p(y|w)\hat{p}(w)dw \\
&= \arg\max_{\beta} \hat{p}(y|\beta) \\
&= \arg\min_{\beta} y'(\lambda I + \Phi\mathrm{B}\Phi')^{-1}y + \log|\lambda I + \Phi\mathrm{B}\Phi'| - 2\sum_{r=1}^{m} \log \varphi_r^{\beta}(\beta_r)
\end{aligned}$$

where $\log \varphi_r^{\beta}(\beta_r)$ is usually set to be a constant for convenience, which in turn makes the true prior, $p(w)$ a Student's t distribution [71], [108].

Obviously, based on the designed criterion, we end with the same ML estimation of hyperparameter β discussed in (4.3.5). The resulting ML problem can be efficiently solved using the Expectation-Maximization (EM) algorithm. The estimation of noise variance λ can also be embedded into the type II maximization (4.5.5) (i.e. $\arg\max_{\beta,\lambda} \hat{p}(y|\beta,\lambda)$) and updated within each iteration of the EM algorithm.

Once $\hat{\beta}$ is determined, we achieve an approximate posterior distribution $\hat{p}(w|y;\hat{\beta})$, from which we can easily calculate its mean $\mu_{\hat{\beta}}$ as the estimate of w and evaluate the confidence around it (e.g. $\int_{\mu_{\hat{\beta}}-\epsilon}^{\mu_{\hat{\beta}}+\epsilon}\mathcal{N}(w|\mu_{\hat{\beta}},\Sigma_{\hat{\beta}})\,dw$). The estimate, $\hat{w}=\mu_{\hat{\beta}}$ is sparse if $\hat{\beta}$ is sparse. Moreover, $\hat{w}$ and $\hat{\beta}$ share the same sparsity pattern.

SBL estimates w by solving a dual problem (4.5.5) in a hyperparameter space. Therefore, it is not clear how SBL enforces sparse solutions. This question cannot be simply explained by the introduced sparse inducing prior as it is approximated by its lower bound in the algorithm. To answer the question, it is necessary to retrieve the estimation of w back to the original parameter space.

First, note that $y'(\lambda I+\Phi\mathrm{B}\Phi')^{-1}y=\min_{w}\lambda^{-1}\|y-\Phi w\|_2^2+w'\mathrm{B}^{-1}w$. By substituting it into (4.5.5), we have:

(4.5.6)

$$\min_{w}\|y-\Phi w\|_2^2+\lambda g_{SBL}(w)$$

where

(4.5.7)

$$g_{SBL}(w)=\min_{\beta}w'\mathrm{B}^{-1}w+\log|\lambda I+\Phi\mathrm{B}\Phi'|-2\sum_{r=1}^{m}\log\varphi_r^{\beta}(\beta_r)$$

It is easy to see that the solution, $\hat{w}$ to (4.5.6) equates to $\mu_{\hat{\beta}}$.

According to (4.5.6), SBL is equivalent to introducing a prior $p(w)\propto\exp\left[-\frac{1}{2}g_{SBL}(w)\right]$ and then solving a canonical MAP problem. Nevertheless, such a prior is non-factorial in contrast to that of traditional MAP methods (discussed in section 4.2.1). In other words, elements of parameter vector w are correlated due to non-factorial priors rather than independent if priors are factorial. As indicated by (4.5.7), the correlation is related to the predefined dictionary matrix, Φ and is also influenced by the noise variance. As a result, the prior is adjusted with respect to measured data.

More importantly, it has been shown in [108] that the penalty function, $g_{SBL}(w)$ is equivalent to:

(4.5.8)

$$g_{SBL}(w)=\min_{z\geq 0}\sum_{r=1}^{m}2\sqrt{z_r}|w_r|-h^*(z)$$

where $h^*(z)$ is the concave conjugate of $\log|\lambda I+\Phi\mathrm{B}\Phi'|$.

As a result, $g_{SBL}(w)$ is permutation invariant, sign invariant and concave on the positive orthant (as it is the minimum of a family of affine functions of w). Therefore, the conditions of Theorem 5.2.1 are satisfied and SBL is guaranteed to produce sparse solutions. In addition, as the noise variance approaches 0, the global minimum of SBL is the maximally sparse solution to the regression problem and no MAP problem with separable penalties (factorial priors) can have fewer local optima than SBL if some conditions on the dictionary matrix are met [108]. This statement implies great advantages of SBL over normal MAP methods.

Remark 4.5.1: The optimization problem (4.5.5) of SBL can also be derived using a sparse

inducing prior represented in the integral type variational form and then following the framework of empirical Bayes (see [71]). Nevertheless, the criterion to measure the gap between the true posterior distribution and its approximation is implicit in this case. Therefore, we prefer to discuss SBL with priors represented in the convex type variational form.

Fortunately, system identification problems of many mathematical models can be recast as a linear regression problem. As a result, SBL can be applied to favor sparse parameter vectors, which in turn imposes sparse network topologies.

Remark 4.5.2: So far, we have only discussed element SBL that imposes element sparsity to parameter vector w. If w is group sparse, group SBL (GSBL) can be applied [169], [170], [148]. GSBL is a minor variant of SBL where each group of elements shares a unique hyperparameter so that elements in a group have the same sparsity pattern.

4.6 Description of sparse linear networks

We use a parametrized multivariable ARX model, $\mathcal{M}^*(w^*)$ to describe a sparse network of p nodes and m inputs [89]:

$$A(q^{-1};w^*)Y(t) = B(q^{-1};w^*)Y(t) + E(t) \tag{4.6.1}$$

where

$$A(q^{-1};w^*) = I + \hat{A}_1 q^{-1} + \cdots + \hat{A}_{n_a^*} q^{-n_a^*}$$
$$B(q^{-1};w^*) = \hat{B}_1 q^{-1} + \cdots + \hat{B}_{n_b^*} q^{-n_b^*}$$

q^{-1} is the time shift operator. $Y(t) \in \mathbb{R}^p$ are the nodes of the network, $U(t) \in \mathbb{R}^m$ denote inputs, and $E(t) \in \mathbb{R}^p$ are i.i.d. white Gaussian noise with a diagonal covariance matrix. n_a^* and n_b^* are polynomial orders (system order). $\hat{A}_i \in \mathbb{R}^{p\times p}$ and $\hat{B}_i \in \mathbb{R}^{p\times m}$ are polynomial coefficients. They are contained in the parameter vector, w^*. $A(q^{-1};w^*)$ is a polynomial matrix that implies the connectivity among nodes including self-loops. Similarly, $B(q^{-1};w^*)$ is a polynomial matrix that decides how the inputs control the nodes. The topology of the network is reflected by the non-zero elements in $A(q^{-1};w^*)$ and $B(q^{-1};w^*)$ whereas system dynamics are dominated by the input-output map of the model [171]:

$$Y(t) = G_u(q^{-1};w^*)U(t) + G_e(q^{-1};w^*)E(t) \tag{4.6.2}$$

where $G^*(q^{-1};w^*)$ denotes the transfer matrix of the model, $\mathcal{M}^*(w^*)$:

$$\begin{aligned} G_u(q^{-1};w^*) &= A^{-1}(q^{-1};w^*)B(q^{-1};w^*) \\ G_e(q^{-1};w^*) &= A^{-1}(q^{-1};w^*) \\ G^*(q^{-1};w^*) &= [G_u(q^{-1};w^*) \quad G_e(q^{-1};w^*)] \end{aligned} \tag{4.6.3}$$

It is shown in [89] that multivariable ARX models are strictly globally identifiable with fixed

system order. In practice, the true system order is unknown. For ARX model (4.6.1), we construct a model set of ARX, $\mathcal{M}(w)$ that contains the ground truth model ($\exists w^{\star}: G(q^{-1}; w^{\star}) = G^{*}(q^{-1}; w^{*})$) but with much higher model complexity. The system dynamics of $\mathcal{M}(w^{\star})$ are the same with $\mathcal{M}^{*}(w^{*})$. $\mathcal{M}(w^{\star})$ is parametrized in a way that the coefficients of the excessive polynomial terms compared to $\mathcal{M}^{*}(w^{*})$ are 0 while the remainings are equal to w^{*}. It should be noticed that $w^{\star}$ is unique with respect to w^{*} and $\mathcal{M}(w^{\star})$ is globally identifiable. As a result, the identification problem of the model, $\mathcal{M}(w^{\star})$ is well-posed. Consequently, the estimation of polynomial orders of $\mathcal{M}^{*}(w^{*})$ is converted into the selection of non-zero model parameters of $\mathcal{M}(w)$. With the estimated model, $\mathcal{M}(w^{\star})$, the polynomial order of $\mathcal{M}^{*}(w^{*})$ is indicated by the highest order of non-zero polynomial terms and model parameters of $\mathcal{M}^{*}(w^{*})$ are shown by non-zero polynomial coefficients.

4.7 Inference problem of ARX-based networks

Given the postulated model, we would like to formulate its identification as a linear regression problem and see how the sparse topology of the network and model complexity are reflected by the structure of the parameter vector.

To guarantee the ground truth model, $\mathcal{M}^{*}(w^{*})$ is contained in the model set, $\mathcal{M}(w)$, the polynomial order, k is set sufficiently large. We parameterize subsystems for each node in the same way. For the ith node:

(4.7.1)

$$\begin{aligned} y_i(t) = &-[A(q^{-1})]_{i1}y_1(t) - \cdots + \{1 - [A(q^{-1})]_{ii}\}y_i(t) \\ &+[B(q^{-1})]_{i1}u_1(t) + \cdots + [B(q^{-1})]_{im}u_m(t) + e_i(t) \end{aligned}$$

$y_j(t)$ denotes the jth node, $u_j(t)$ the jth input, $e_i(t)$ i.i.d. Gaussian noise and:

$$\begin{aligned} [A(q^{-1})]_{ii} &= a_1^{ii}q^{-k} + a_2^{ii}q^{-k+1} + \cdots + a_k^{ii}q^{-1} + 1 \\ [A(q^{-1})]_{ij} &= a_1^{ij}q^{-k} + \cdots + a_{k-1}^{ij}q^{-2} + a_k^{ij}q^{-1} \\ [B(q^{-1})]_{i1} &= b_1^{ij}q^{-k} + \cdots + b_{k-1}^{ij}q^{-2} + b_k^{ij}q^{-1} \\ k &\geq \max\{n_a^*, n_b^*\} \end{aligned}$$

where a and b denote coefficients of polynomial terms. Hereafter, w is a vector that only includes the parameters of subsystem (4.7.1) of the ith node.

Assume the availability of time series data collected from discrete time indices 1 to t for each node and input. For the ith node, we define the following matrices and vectors:

(4.7.2)

$$\begin{aligned} y = \begin{bmatrix} y_i(t) \\ \vdots \\ y_i(k+1) \end{bmatrix},\ w = \begin{bmatrix} w_1 \\ \vdots \\ w_{p+m} \end{bmatrix} \\ \Phi = \left[\Phi_1 \middle| \cdots \middle| \Phi_{p+m}\right] \\ \lambda = E\{e_i(t)^2\},\ E\{e_i(t)\} = 0 \end{aligned}$$

$$w_r = \begin{cases} \begin{bmatrix} a_1^{ir} & \cdots & a_k^{ir} \end{bmatrix}' & \text{if } 1 \le r \le p \\ \begin{bmatrix} b_1^{i(r-p+1)} & \cdots & b_k^{i(r-p+1)} \end{bmatrix}' & \text{if } p < r \le p+m \end{cases}$$

$$\Phi_{1\le r\le p} = \begin{bmatrix} -y_r(t-k:t-1) \\ \vdots \\ -y_r(1:k) \end{bmatrix},\ \Phi_{p<r\le p+m} = \begin{bmatrix} u_{r-p}(t-k:t-1) \\ \vdots \\ u_{r-p}(1:k) \end{bmatrix}$$

where $y \in \mathbb{R}^{t-k}$, $w \in \mathbb{R}^{k(p+m)}$ and $\Phi \in \mathbb{R}^{(t-k)\times k(p+m)}$.

Vector w is divided into $p+m$ groups, each of which contains polynomial coefficients with respect to a specific node or input. For example, w_3 consists of coefficients of $[A(z^{-1})]_{i3}$ of node 3. Elements within each group are indexed so that w_{rj} denotes the jth coefficient, a_j^{ir} of $[A(z^{-1})]_{ir}$. It should be noticed that if node or input r does not control node i, group w_r or w_{r+p} equates to 0. In addition, within each group, coefficients of excessive polynomial terms are 0 ($w_{rj} = 0$ if $r \le p$, $j > n_a^*$ or $r > p$, $j > n_b^*$). Consequently, w is both group and element sparse. The group sparsity pattern of estimated w indicates the network topology while the zero structure within each group implies polynomial orders.

With the defined matrices and vectors, we have the following model:

(4.7.3)

$$y = \Phi w + \varepsilon$$

where $\varepsilon = [e_i(t)\ \ e_i(t-1)\cdots e_i(k+1)]'$.

Hence, the identification problem is equivalent to estimating an element and group sparse parameter vector of a linear regression model.

The likelihood function based on Bayes' rules is:

(4.7.4)

$$p(y|w,\lambda) = \frac{1}{(2\pi\sigma^2)^{\frac{t-k}{2}}} exp\left(-\frac{1}{2\lambda}\|y-\Phi w\|_2^2\right)$$

where the conditioning on the inputs and other nodes is suppressed for simplicity. It should be noticed that dictionary matrix Φ is correlated with the observed data, y. Hence, the likelihood is not trivially Gaussian.

Remark 4.7.1: For a homogeneous dataset which is collected from multiple experiments subject to the same experimental conditions, the likelihood function is the product of that of individual experiment. In contrast, given a heterogeneous dataset, the problem can be formulated in a similar way where some groups of w share the same sparsity pattern [172]. Hence, our framework can still be applied to deal with heterogeneous datasets with minor modifications.

Although likelihood function (4.7.4) is not presented in a standard Gaussian form, its logarithm is a quadratic function with respect to w. By maximizing the likelihood, we end with the PEM method. With infinite data points, PEM is guaranteed to converge to the ground truth model [89]. In practice, given limited data, PEM may suffer from over-fitting. In addition, coefficients of the excessive polynomials of the estimate remain non-zero and the resulting

network is full-connected. In other words, the estimated w is not sparse. Therefore, penalties for both network topology and model complexity are essential. Referring to (4.7.2), a sparse network causes a group sparse w, whereas sparsity within each group indicates reduced order of polynomials. A direct framework of MAP to achieve those two kinds of sparsity is Sparse Group Lasso (SGL) [143]. Nevertheless, we resort to SBL in order to simplify the tuning process and achieve better performance.

Remark 4.7.2: In practice, only a few networks can be sufficiently interpreted by ARX models whereas nonlinear ARX models (NARX) are a better choice. NARX has been widely applied in control engineering [173], systems biology [110] and machine learning [174]. We will see that our framework can be directly extended to NARX. We will discuss this issue in the following sections.

4.8 Combined group and element sparse Bayesian learning

4.8.1 Sparsity inducing priors

Full Bayesian treatment requires introducing a prior distribution for w. We define a distribution $p(w)$ in a general form as: $p(w) \propto \exp\left[-\frac{1}{2}\sum_j g(w_j)\right]$. Since w is sparse, a sparse inducing prior like Generalized Gaussian, Student's t or Logistic is assigned to $p(w)$. However, estimating w as the posterior mean is intractable because the posterior distribution, $p(w|y)$ is non-Gaussian and not analytical.

To tackle the problem, we apply SBL as a deterministic approximation scheme. As discussed, parameter vector w is both element and group sparse. There are priors able to induce either of these two types of sparsity. We use these priors to construct a novel one that can impose both of them at the same time.

Priors that are able to induce element sparsity to $w \in \mathbb{R}^{k(p+m)}$ can be expressed in the convex type variational form as [68], [175]:

(4.8.1)

$$p(w) = \prod_{r=1}^{p+m} p(w_r) = \max_{\beta \geq 0} \mathcal{N}(w|\mathbf{0}, B)\varphi^{\beta}(\beta)$$

$$p(w_r) = \prod_{j=1}^{k} p(w_{rj}) = \max_{\beta_r \geq 0} \mathcal{N}(w_r|\mathbf{0}, B_r)\, \varphi_r^{\beta}(\beta_r)$$

$$p(w_{rj}) = \max_{\beta_{rj} \geq 0} \mathcal{N}(w_{rj}|0, \beta_{rj})\, \varphi_{rj}^{\beta}(\beta_{rj})$$

where subscript r denotes the rth group in a vector and subscript j the jth element in that group. $\beta = vec\{\beta_1, \dots, \beta_{p+m}\} \in \mathbb{R}^{k(p+m)}$ is a vector of hyperparameters and $\beta_r = vec\{\beta_{r1}, \dots, \beta_{rk}\} \in \mathbb{R}^k$. B is the covariance matrix of a Gaussian distribution and is parameterized by vector β as $B = blkdiag\{B_1, \dots, B_{p+m}\}$ and $B_r = diag\{\beta_r\}$. $\varphi_{rj}^{\beta}(\beta_{rj})$ is a positive function that depends on the prior, $p(w)$. $\varphi_r^{\beta}(\beta_r) = \prod_{j=1}^{k} \varphi_{rj}^{\beta}(\beta_{rj})$ and

$\varphi^{\beta}(\beta) = \prod_{r=1}^{p+m} \varphi_r^{\beta}(\beta_r)$.

To impose group sparsity, the hyperparameters of each group are unified so that elements in a group share the same sparsity pattern [110], [169]:

(4.8.2)

$$p(w) = \prod_{r=1}^{p+m} p(w_r) = \max_{\gamma \geq 0} \mathcal{N}(w|\mathbf{0}, \Gamma)\varphi^{\gamma}(\gamma)$$

$$p(w_r) = \max_{\gamma_r \geq 0} \mathcal{N}(w_r|\mathbf{0}, \gamma_r I)\, \varphi_r^{\gamma}(\gamma_r)$$

where $\gamma = vec\{\gamma_1, \dots, \gamma_{p+m}\}\mathbb{R}^{p+m}$ is a vector of hyperparameters and $\Gamma = blkdiag\{\gamma_1 I, \dots, \gamma_{p+m} I\}$. $\varphi_r^{\gamma}(\gamma_r)$ is a positive function and $\varphi^{\gamma}(\gamma) = \prod_{r=1}^{k} \varphi_r^{\gamma}(\gamma_r)$.

Remark 4.8.1: In practice, the group of w that presents auto-regression of ARX can be excluded from the group sparsity inducing prior to improve the estimation accuracy.

According to the expression of element and group sparsity inducing priors, neither of them is suitable to impose both kinds of sparsity. The hyperparameters in (4.8.1) are independent so that the resulting lower bound $\hat{p}(w|\beta)$ exhibits no correlations among components within each group. Hence, (4.8.1) is powerless to impose group sparsity. In contrast, the lower bound $\hat{p}(w|\gamma)$ of (4.8.2) prohibits element sparsity within each group because of the shared hyperparameter, thus too rigid. To promote element and group sparsity at the same time, we combine (4.8.1) and (4.8.2) to deduce a new distribution:

(4.8.3)

$$p(w) = C \max_{\beta \geq 0, \gamma \geq 0} \mathcal{N}(w|\varepsilon, \mathrm{B})\mathcal{N}(w|\mathbf{0}, \Gamma)\varphi^{\beta}(\beta)\varphi^{\gamma}(\gamma)$$

where C is the normalization constant that can be absorbed by positive functions $\varphi^{\beta}(\beta)$ or $\varphi^{\gamma}(\gamma)$ and is independent on hyperparameters, β and γ. ε is the expected value of w. As w is element sparse within each non-zero group, ε is set close to $\mathbf{0}$ ($\|\varepsilon\| \approx 10^{-3}$). Hence, we achieve an improper prior as the lower bound of the original one:

(4.8.4)

$$\begin{aligned} \hat{p}(w) &= \mathcal{N}(w|\varepsilon, \mathrm{B})\mathcal{N}(w|\mathbf{0}, \Gamma)\varphi^{\beta}(\beta)\varphi^{\gamma}(\gamma) \\ &\leq p(w) \end{aligned}$$

The prior in (4.8.4) shows that two types of sparsity are controlled by two series of hyperparameters, β and γ, respectively. As γ_r approaches 0, the rth group of w is enforced to 0 regardless of β_r. That means the group sparsity can be determined from a hyperparameter space of dimension $p + m$ instead of $k(p + m)$ if only element sparsity inducing priors are applied. Furthermore, within a non-zero group ($\gamma_r > 0$), hyperparameter $\beta_r \in \mathbb{R}^k$ enables extra freedom to characterize the elementary values of w_r while GSBL only allows one degree of freedom via $\gamma_r \in \mathbb{R}$. The value of these hyperparameters is unknown and remains to be determined using observed data.

Remark 4.8.2: The conventional way to promote element and group sparsity is to use

hierarchical Bayesian by introducing two hyperparameters for group and element sparsity independently, where one hyperparameter is conditioned on the other. However, the hyperparameter which is deeper in the hierarchy has a weaker impact on the inference procedure [163]. This means that the resulting penalty is not able to impose both group and element sparsity. As such, multiplying two priors makes sense since both hyperparameters influence w directly.

4.8.2 Bayesian approximation to posterior distribution

Although prior $\hat{p}(w)$ is improper, we can still deduce a normalized posterior distribution of w as:

(4.8.5)

$$\hat{p}(w|y) = \frac{p(y|w)\hat{p}(w)}{\int p(y|w)\hat{p}(w)dw}$$

Clearly, $\hat{p}(w|y)$ is Gaussian since $log\,(\hat{p}(w|y))$ is a quadratic function of w:

(4.8.6)

$$\hat{p}(w|y) \sim \mathcal{N}(w|\mu, \Sigma)$$

where

$$\Sigma = [(\Gamma^{-1} + B^{-1}) + \lambda^{-1}\Phi'\Phi]^{-1}$$
$$\mu = \Sigma(\lambda^{-1}\Phi' y - B^{-1}\varepsilon)$$

The sparsity of the estimated w (i.e. $E(w|y) = \mu$) depends on hyperparameters β and γ. They are optimized using (4.5.5), which is called evidence maximization or type II maximization [90], [107]:

(4.8.7)

$$\begin{aligned}(\beta^*, \gamma^*, \lambda^*) &= \arg\min_{\beta,\gamma,\lambda\geq 0} \int p(y|w)|p(w) - \hat{p}(w)|dw \\ &= \arg\min_{\beta,\gamma,\lambda\geq 0} -2\log \int p(y|w)\hat{p}(w)dw \\ &= \arg\min_{\beta,\gamma,\lambda\geq 0} -2\log \hat{p}\,(y|\beta,\gamma,\lambda)\end{aligned}$$

Remark 4.8.3: It should be noticed that not all the sparsity inducing priors can lead to a sparse solution under the framework of SBL. That is, the selection of functions $\varphi^{\beta}(\beta)$ and $\varphi^{\gamma}(\gamma)$ influences the sparsity pattern of the final result. It was shown that one reasonable choice is that $-log\varphi(\cdot)$ is concave and non-decreasing [108]. For simplicity, we set $\varphi(\cdot)$ as a constant. Therefore, it can be ignored in the following discussions.

In fact, the estimation of w can be embedded into problem (4.8.7) as follows:

Proposition 4.8.1: The estimation of w as the posterior mean in (4.8.6) can be embedded into the type II maximization (4.8.7), thus resulting in the optimization problem:

$$\mathcal{L}\colon \min_{w,\beta,\gamma,\lambda} \lambda^{-1}\|y - \Phi w\|_2^2 + \|w\|_{\Gamma^{-1}}^2 + \|w - \varepsilon\|_{\mathrm{B}^{-1}}^2$$

$$+\log|\mathrm{B}+\Gamma|+\log|\lambda I+\Phi(\Gamma^{-1}+\mathrm{B}^{-1})^{-1}\Phi'|$$

subject to:

$$\beta\geq 0,\gamma\geq 0,\lambda\geq 0$$

Proof:

First, note that:

(A1)

$$-2\log\int p(y|w)\hat{p}(w)dw$$

$$=-2\log\int\frac{1}{(2\pi\sigma^2)^{\frac{t-k}{2}}}exp\left(-\frac{1}{2\lambda}\|y-\Phi w\|_2^2\right)\mathcal{N}(w|\varepsilon,\mathrm{B})\mathcal{N}(w|\mathbf{0},\Gamma)dw$$

$$=-2\log\int\exp(E_w)\,dw+\log|\lambda I|+\log|\mathrm{B}|+\log|\Gamma|$$

where by ignoring all the constant terms, we have:

$$E_w=-\frac{1}{2}\left[\lambda^{-1}\|y-\Phi w\|_2^2+\|w\|_{\Gamma^{-1}}^2+\|w-\varepsilon\|_{\mathrm{B}^{-1}}^2\right]$$

By completing the squares:

(A2)

$$-\frac{1}{2}\left[\lambda^{-1}\|y-\Phi w\|_2^2+\|w\|_{\Gamma^{-1}}^2+\|w-\varepsilon\|_{\mathrm{B}^{-1}}^2\right]$$

$$=-\frac{1}{2}\left[(w-\mu)'\Sigma^{-1}(w-\mu)+E_y\right]$$

where

$$\Sigma=[(\Gamma^{-1}+B^{-1})+\lambda^{-1}\Phi'\Phi]^{-1}$$
$$\mu=\Sigma(\lambda^{-1}\Phi'y-B^{-1}\varepsilon)$$

$$E_y=\min_w\lambda^{-1}\|y-\Phi w\|_2^2+\|w\|_{\Gamma^{-1}}^2+\|w-\varepsilon\|_{\mathrm{B}^{-1}}^2$$

In addition, note that:

(A3)

$$-2\log\int\exp\left\{-\frac{1}{2}[(w-\mu)'\Sigma^{-1}(w-\mu)]\right\}dw$$
$$=-\log|\Sigma|$$
$$=\log|(\Gamma^{-1}+B^{-1})+\lambda^{-1}\Phi'\Phi|$$

Using (A2) and (A3), the integral (A1) becomes:

$$-2\log\int p(y|w)\hat{p}(w)dw$$
$$=E_y+\log|(\Gamma^{-1}+B^{-1})+\lambda^{-1}\Phi'\Phi|+\log|\lambda I|+\log|\mathrm{B}|+\log|\Gamma|$$
$$=E_y+\log|I+(\Gamma^{-1}+B^{-1})^{-1}\lambda^{-1}\Phi'\Phi|+\log|\lambda I|+\log|\mathrm{B}+\Gamma|$$
$$=E_y+\log|I+\lambda^{-1}\Phi(\Gamma^{-1}+B^{-1})^{-1}\Phi'|+\log|\lambda I|+\log|\mathrm{B}+\Gamma|$$
$$=E_y+\log|\lambda I+\Phi(\Gamma^{-1}+B^{-1})^{-1}\Phi'|+\log|\mathrm{B}+\Gamma|$$

Consequently, we end with:

$$\mathcal{L}:\min_{w,\beta,\gamma,\lambda}\lambda^{-1}\|y-\Phi w\|_2^2+\|w\|_{\Gamma^{-1}}^2+\|w-\varepsilon\|_{\mathrm{B}^{-1}}^2$$

$$+\log|\mathrm{B}+\Gamma|+\log|\lambda I+\Phi(\Gamma^{-1}+\mathrm{B}^{-1})^{-1}\Phi'|$$

subject to:

$$\beta\geq 0,\gamma\geq 0,\lambda\geq 0$$

4.9 Algorithms to solve Type II maximization

The optimization problem $\mathcal{L}$ is nonlinear and non-convex. However, the cost function has some good properties, which encourage the usage of more advanced algorithms other than traditional gradient descent methods. It turns out that the cost function can be decomposed into a difference of two convex terms. Therefore, the optimization problem is solved as a Difference of Convex Programming (DCP). In addition, we prefer to employ distributed algorithms because, in practice, the inference problem often has to deal with large-scale networks and enormous datasets. We find that the resulting DCP can be further decomposed using Alternating Direction Method of Multipliers (ADMM) which makes it possible to tackle large-scale networks and utilize limited computational power more efficiently.

4.9.1 Convex-Concave Procedure

The cost function is separated into two parts in the following way:

(4.9.1)

$$\mathcal{L}:\min_{w,\beta,\gamma,\lambda} u(w,\beta,\gamma,\lambda)-v(\beta,\gamma,\lambda)$$

where

$$u(w,\beta,\gamma,\lambda)=\lambda^{-1}\|y-\Phi w\|_2^2+\|w\|_{\Gamma^{-1}}^2+\|w-\varepsilon\|_{\mathrm{B}^{-1}}^2$$
$$v(\beta,\gamma,\lambda)=-\log|\mathrm{B}+\Gamma|-\log|\lambda I+\Phi(\Gamma^{-1}+\mathrm{B}^{-1})^{-1}\Phi'|$$

Proposition 4.9.1: Functions $u(w,\beta,\gamma,\lambda)$ and $v(\beta,\gamma,\lambda)$ are both jointly convex with respect to their own variables.

Proof:

To see functions $u(w,\beta,\gamma,\lambda)$ and $v(\beta,\gamma,\lambda)$ are convex, we need to prove each term of these functions is jointly convex. To check the convexity of $u(w,\beta,\gamma,\lambda)$, we consider the epigraph of its terms defined as $epi\, f=\{(x,t)|x\in dom\, f, f(x)<t\}$ [150]. It is known that:

$$\lambda I\succ 0,\ \lambda^{-1}\|y-\Phi w\|_2^2<t \text{ is equivalent to}$$
$$\begin{bmatrix}\lambda I & y-\Phi w\\ (y-\Phi w)' & t\end{bmatrix}\succ 0$$

Hence, the term $\lambda^{-1}\|y-\Phi w\|_2^2$ is jointly convex as is same with $\|w\|_{\Gamma^{-1}}^2$ and $\|w-\varepsilon\|_{\mathrm{B}^{-1}}^2$ since their epigraphs are convex sets described by LMIs.

For the function $v(\beta,\gamma,\lambda)$, first note that $-\log|\cdot|$ is a convex function in S^+. Since $\mathrm{B}+\Gamma$ is an affine function of B and Γ, $-\log|\mathrm{B}+\Gamma|$ is jointly convex with respect to B and Γ. The second term of the function seems more complex. To prove its convexity, we first consider

the following lemma. A similar lemma with lower dimension of function domain can be found in [150].

Lemma: Suppose a function $f(x) = h(g_1(x), \dots, g_k(x))$ where $h(z_1, \dots, z_k): \mathbb{R}^k \to \mathbb{R}$ and $g_r: \mathbb{R}^n \to \mathbb{R}$, then $f(\cdot)$ is concave if $h(\cdot)$ is concave and non-decreasing in each argument and g_r is concave.

The proof of this lemma is straightforward by checking the Hessian matrix:

$$\nabla^2 f = J_g{}' \nabla^2 h J_g + \sum_{l=1}^{k} \frac{\partial h}{\partial z_l}|_{g_l} \nabla^2 g_l$$

where J_g denotes the Jacobian matrix of the function $g = [g_1, \dots, g_k]'$.

Let $h(x^1, x) = \log|x^1 I + \Phi diag(x)\Phi'|$ and $g^1(\beta, \gamma, \lambda) = \lambda$, $g_{rj}(\beta, \gamma, \lambda) = \frac{\gamma_r \beta_{rj}}{\gamma_r + \beta_{rj}}$ with $x \in \mathbb{R}^{k(p+m)}$, $r \in [1, p+m]$ and $j \in [1, k]$, then $h\left(g^1, g_{11}, g_{12}, \dots, g_{(p+m)k}\right) = \log|\lambda I + \Phi(\Gamma^{-1} + \mathrm{B}^{-1})^{-1}\Phi'|$. Obviously, $h(\cdot)$ is jointly concave with respect to x^1 and x.

Since the Hessian matrix of g^1 is 0, we only need to check the gradient of $h(\cdot)$ with respect to x:

$$\begin{aligned} \frac{\partial h}{\partial x_r} &= trace\{\Phi'(x^1 I + \Phi diag(x)\Phi')^{-1}\Phi diag(e_r)\} \\ &= \Delta_{rr} \end{aligned}$$

where $\Delta = \Phi'(x^1 I + \Phi diag(x)\Phi')^{-1}\Phi$ and e_r is a vector with its rth element 1 and all the others 0.

Since matrix Δ is at least semi-positive definite, its diagonal elements must be non-negative. As a result, $h(\cdot)$ is non-decreasing in each argument. We finally calculate the Hessian matrix, H of $g_{rj}(\beta, \gamma, \lambda)$. Note that matrix entries $H_{qq} = \frac{\partial^2 g_{rj}}{\partial \beta_{rj}^2}$, $H_{ll} = \frac{\partial^2 g_{rj}}{\partial \gamma_r^2}$ and $H_{ql} = H_{lq} = \frac{\partial^2 g_{rj}}{\partial \beta_{rj} \partial \gamma_r}$ with all the other entries 0, where $q = (r-1)k + j$ and $l = (p+m)k + r$. It is always possible to find a permutation matrix, P such that:

$$P'HP = \begin{bmatrix} \widehat{H} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

where

$$\widehat{H} = \begin{bmatrix} -\frac{2\gamma_r^2\left(\gamma_r + \beta_{rj}\right)}{\left(\gamma_r + \beta_{rj}\right)^4} & \frac{2\gamma_r\beta_{rj}\left(\gamma_r + \beta_{rj}\right)}{\left(\gamma_r + \beta_{rj}\right)^4} \\ \frac{2\gamma_r\beta_{rj}\left(\gamma_r + \beta_{rj}\right)}{\left(\gamma_r + \beta_{rj}\right)^4} & -\frac{2\beta_{rj}^2\left(\gamma_r + \beta_{rj}\right)}{\left(\gamma_r + \beta_{rj}\right)^4} \end{bmatrix}$$

Obviously, matrix $\widehat{H}$ is semi-negative definite so that H is also semi-negative definite, which indicates function $g_{rj}(\beta, \gamma, \lambda)$ is concave. For g^1, it is an affine function. According to the lemma above, $\log|\lambda I + \Phi(\Gamma^{-1} + \mathrm{B}^{-1})^{-1}\Phi'|$ is jointly concave with respect to β, γ

and λ.

Now, the optimization problem is transferred into a difference of convex programming (DCP). It can be solved using sequential convex optimization techniques. Here, we use convex-concave procedure (CCCP) which is a kind of majorization-minimization (MM) algorithms using the linear majorization function [91], [176]. For $\min_x f(x)$ where $f(x) = u(x) - v(x)$, and $u(x)$ and $v(x)$ are convex, we can solve it iteratively by:

(4.9.2)

$$x^{n+1} = \arg\min_x u(x) - \langle x, \nabla v(x^n) \rangle$$

where $\langle \cdot, \cdot \rangle$ denotes inner product.

Therefore, in each iteration, we solve a convex problem:

(4.9.3)

$$(w^{n+1}, \beta^{n+1}, \gamma^{n+1}, \lambda^{n+1})$$
$$= \arg\min_{w,\beta,\gamma,\lambda} u(w,\beta,\gamma,\lambda) - \frac{\partial v}{\partial \lambda}|_{\beta^n,\gamma^n,\lambda^n}\lambda - \nabla'_\beta v|_{\beta^n,\gamma^n,\lambda^n}\beta - \nabla'_\gamma v|_{\beta^n,\gamma^n,\lambda^n}\gamma$$

subject to

$$\beta \geq 0, \gamma \geq 0, \lambda \geq 0$$

where

$$-\frac{\partial v}{\partial \lambda} = trace\{\Delta\}$$
$$-[\nabla_\beta v]_{rj} = (\beta_{rj} + \gamma_r)^{-1} + \frac{\gamma_r^2 [\Phi' \Delta \Phi]_{qq}}{(\beta_{rj} + \gamma_r)^2}$$
$$-\frac{\partial v}{\partial \gamma_r} = \sum_{j=1}^{k} \left(\frac{1}{\gamma_r + \beta_{rj}} + \frac{\beta_{rj}^2 [\Phi' \Delta \Phi]_{qq}}{(\gamma_r + \beta_{rj})^2} \right)$$
$$\Delta = [\lambda I + \Phi(\Gamma^{-1} + \mathrm{B}^{-1})^{-1}\Phi']^{-1}$$
$$q = (r-1)k + j$$

By optimizing β, γ and λ first, we get analytical expressions for their optimal solutions as functions of w:

(4.9.4)

$$\beta_{rj}^{opt} = \frac{|w_{rj} - \varepsilon_{rj}|}{\sqrt{g_{rj}^\beta}}, \quad \gamma_{rj}^{opt} = \frac{\|w_r\|}{\sqrt{g_r^\gamma}}, \quad \lambda^{opt} = \frac{\|y - \Phi w\|_2}{\sqrt{g^\lambda}}$$

where

$$g_{rj}^\beta = -[\nabla v_\beta|_{\beta^n,\gamma^n,\lambda^n}]_{rj}$$

$$g_r^\gamma = -[\nabla v_\gamma|_{\beta^n,\gamma^n,\lambda^n}]_r$$

$$g^\lambda = -\frac{\partial v}{\partial \lambda}|_{\beta^n,\gamma^n,\lambda^n}$$

It is easy to see that solutions (4.9.2) are feasible since they are non-negative. Therefore, sub-problem (4.9.3) can be further simplified by substituting (4.9.4):

$$w^{n+1} = \arg\min_{w} \sqrt{g^{\lambda}}\|y - \Phi w\|_2 + \sum_{r=1}^{p+m}\left(\sqrt{g_r^{\gamma}}\|w_r\|_2 + \sum_{j=1}^{k}\sqrt{g_{rj}^{\beta}}|w_{rj} - \varepsilon_{rj}|\right) \tag{4.9.5}$$

The optimization (4.9.5) can be solved as a Second Order Cone Program (SOCP). It is a reweighted LASSO type problem with some minor variants. The first term of the data-fitting error is measured by ℓ_2-norm rather than by sum of squares in (4.2.8). Without estimating noise variance λ, (4.9.5) can be reformulated into a standard reweighted SGL problem. In addition, the second term of (4.9.5) blends ℓ_1 and ℓ_2 norms to impose element and group sparsity of w at the same time. The weights of these two terms are updated automatically in each iteration.

A special attention should be paid to the estimation of noise variance λ [71], [170]. Under some circumstances, λ can be compensated by hyperparameters thus unidentifiable. The risk decreases as more data points are available for identification. The estimated noise variance has a significant impact on the final result [169]. If the algorithm fails to produce a reasonable estimate of λ, other schemes should be considered. One traditional method is cross validation. In this way, λ is not regarded as the noise variance but rather a tuning variable. Another approach is to fit the data first with an ARX model of high order and use the residual error as the approximation of the sampled noise [113]. The empirical noise variance can then be calculated from the error. To summarize, Algorithm 4.9.1 represents the procedure above.

Algorithm 4.9.1 Solve $\mathcal{L}$ using CCCP

1: ***Initialization***: set $(\beta^0, \gamma^0, \lambda^0)$

2: ***Calculate*** g_{rj}^{β}, g_r^{γ} and g^{λ} using (4.9.4)

3: ***For*** $t = 1{:}\,MAX$ do

4: Solve the reweighted convex problem:

$$w^{n+1} = \arg\min_{w} \sqrt{g^{\lambda}}\|y - \Phi w\|_2 + \sum_{r=1}^{p+m}\left(\sqrt{g_r^{\gamma}}\|w_r\|_2 + \sum_{j=1}^{k}\sqrt{g_{rj}^{\beta}}|w_{rj} - \varepsilon_{rj}|\right)$$

5: ***Update*** β^{n+1}, γ^{n+1} and λ^{n+1} using (4.9.4)

6: ***Update*** g_{rj}^{β}, g_r^{γ} and g^{λ} using (4.9.3)

7: ***If*** $\|w^{n+1} - w^n\| \le \epsilon$

8: Break

9: ***end if***

10: ***End for***

4.9.2 Alternating Direction Method of Multipliers (ADMM)

If a network possesses a large number of nodes, leading to high dimensional variables,

solving (4.9.5) directly will be very inefficient or even computationally infeasible due to hardware limitations. In this case, we split the optimization problem into a series of small scale sub-problems to reduce computational burden so that they can be coped with in parallel. It turns out that by splitting the cost function, (4.9.5) can be formulated as a sharing problem and solved using ADMM algorithms [109], [177].

The optimization problem (4.9.5) can be treated as a sharing problem in the form of $\sum f_r(x_r) + g(\sum x_r)$ where x_r denotes the rth group of w, and $g(\cdot)$ and $f_r(\cdot)$ are convex. Rewrite the sharing problem in the ADMM form:

(4.9.6)

$$w^{n+1} = \arg\min_{w} \sqrt{g^{\lambda}} \left\| y - \sum_{r=1}^{p+m} z_i \right\|_2 + \sum_{r=1}^{p+m} \left(\sqrt{g_r^{\gamma}} \|w_r\|_2 + \sum_{j=1}^{k} \sqrt{g_{rj}^{\beta}} |w_{rj} - \varepsilon_{rj}| \right)$$

subject to

$$\Phi_r w_r - z_r = 0$$

where

$$\Phi = \left[\Phi_1 \middle| \cdots \middle| \Phi_{p+m} \right]$$

The scaled form of ADMM becomes [177]:

(4.9.7)

$$\begin{aligned} w_r^{n+1} &= \arg\min_{w_r} \sqrt{g_r^{\gamma}} \|w_r\|_2 + \sum_{j=1}^{k} \sqrt{g_{rj}^{\beta}} |w_{rj} - \varepsilon_{rj}| \\ &\quad + \frac{\rho}{2} \|\phi_r w_r - \phi_r w_r^n + \overline{\Phi w}^n - \bar{z}^n + u^n\|_2^2 \\ \bar{z}^{n+1} &= \arg\min_{\bar{z}} \sqrt{g^{\lambda}} \|y - (p+m)\bar{z}\|_2 + \frac{(p+m)\rho}{2} \|\bar{z} - u^n - \overline{\Phi w}^{n+1}\|_2^2 \\ u^{n+1} &= u^n + \overline{\Phi w}^{n+1} - \bar{z}^{n+1} \end{aligned}$$

where

$$\overline{\Phi w}^n = \frac{1}{p+m} \sum_{r=1}^{p+m} \Phi_r w_r^n$$

The original optimization problem of $w \in \mathbb{R}^{k(p+m)}$ is now decomposed into a series of sub-problems, each of which scales as $w_r \in \mathbb{R}^k$. Hence, the computational burden per iteration is greatly relieved. w-update is a standard SGL problem which can be efficiently solved using the accelerated generalized gradient descent algorithm [143]. $\bar{z}$-update is a group LASSO problem and has analytical solutions. Let $\hat{z} = \bar{z} - \frac{y}{p+m}$, the original $\bar{z}$-update becomes:

(4.9.8)

$$\hat{z}^{n+1} = \arg\min_{\bar{z}} \frac{1}{2} \left\| \bar{z} + \frac{y}{p+m} - u^n - \overline{\Phi w}^{n+1} \right\|_2^2 + \frac{\sqrt{g^{\lambda}}}{\rho} \|\hat{z}\|_2$$

so that

$$\hat{z}^{n+1} = \frac{c}{\|c\|_2}(\|c\|_2 - \frac{\sqrt{g^{\lambda}}}{\rho})_+$$

where

$$c = -\frac{y}{p+m} + u^n + \overline{\Phi w}^{n+1}$$

Note that w-update can be solved in parallel independently. $\bar{z}$ and u-update are then solved in sequence after collecting w-update.

4.9.3 Expectation-Maximization (EM)

The EM method is another traditional technique used to solve (4.8.7). It also belongs to the class of majorization-minimization (MM) methods and is a special case of DCA (Difference of Convex Functions Algorithm). While a DC function has infinite many DC decompositions, the way to decompose the function can greatly influence the performance of an algorithm [178].

To maximize a likelihood function $L(\theta) = log\ p(y|\theta)$, EM implements Expectation (E step) and Maximisation (M step) iteratively. In the E step, the function, $Q(\theta, \theta^n) = E_{x|y,\theta^n}[\log\ p(y,x|\theta)] = \int \log p(y,x|\theta)\, p(x|y,\theta^n)\, dx$ is calculated where x is the unobservable latent random variable. In the M step, the optimization problem, $\theta^{n+1} = \arg\max Q(\theta, \theta^n)$ is solved [179], [90]. The generated sequence, $\{\theta^n\}$ leads to the increased likelihood function ($L(\theta^n) < L(\theta^{n+1})$). In our case, we regard w as the latent variable. Following the standard procedure of EM, the algorithm is described in Algorithm 4.9.2.

Algorithm 4.9.2 Solve $\mathcal{L}$ using EM method

1: ***Initialization***: set $(\beta^0, \gamma^0, \lambda^0)$

3: ***For*** $t = 1: MAX$ do

4: E step: Formulate $p(w|\beta^n, \gamma^n, \lambda^n, y)$

5: M step: solve

$$(\beta^{n+1}, \gamma^{n+1}, \lambda^{n+1}) = \arg\min_{w} E_{w|\beta^n,\gamma^n,\lambda^n}\{\ln p(y, w|\beta, \gamma, \lambda)\}$$

5: ***Update*** solutions of M step as:

$$\gamma_r^{n+1} = \frac{1}{k}\sum_{j=1}^{k}[\Sigma^n]_{qq} + \left(\mu_{rj}^n\right)^2$$

$$\beta_{rj}^{n+1} = [\Sigma^n]_{qq} + \left(\mu_{rj}^n - \varepsilon_{rj}\right)^2$$

$$\lambda^{n+1} = \frac{\|y - \Phi\mu\|_2^2 + \lambda^n \sum_{r=1}^{p+m}\sum_{j=1}^{k} 1 - \tau_{rj}[\Sigma^n]_{qq}}{N}$$

where

$$\tau_{rj} = \left(\beta_{rj}^n\right)^{-1} + (\gamma_r^n)^{-1},\ q = (r-1)k + j,\ N = k(p+m)$$

7: ***If*** $\|\mu^{n+1} - \mu^n\| \le \epsilon$

8: Break

9: ***end if***

10: ***End for***

The cost of EM per iteration is dominated by the inversion of covariance matrix Σ in (4.8.6). At first glance, the required work seems computationally demanding ($O[k^3(p+m)^3]$). Nevertheless, by applying the Sherman-Morrison-Woodbury formula, the cost is reduced to $O([k(p+m)(t-k)^2])$. Consequently, the total cost per iteration is $O([k(p+m)(t-k)^2])$ assuming the scale of the target network is large and the measured data are limited ($t-k \ll k(p+m)$).

4.10 Extension to nonlinear ARX models

In reality, most networks are nonlinear (e.g. genetic regulation networks). To cope with this fact, we extend our method to nonlinear ARX models (NARX) whose nonlinear terms are linear in parameters. As a grey-box model, a NARX is built based on a predefined dictionary of basis functions. Its identification is normally formulated as a sparse basis selection problem [111], [180].

Consider a network with p nodes and m inputs: $A(q^{-1};w)Y(t) = B(q^{-1};w)U(t) + F(t;w) + E(t)$, where the form of matrices $A(q^{-1};w)$ and $B(q^{-1};w)$ is exactly the same with (4.6.1). $F(t;w)$ is a vector of nonlinear functions depending on the past values of nodes and inputs. Each element of $F(t;w)$ is a linear combination of basis functions. The topology of the network is reflected by the non-zero elements in $A(q^{-1};w)$, $B(q^{-1};w)$, and nonlinear terms of $F(t;w)$ whereas the system dynamics are dominated by the elements in these matrices.

We parametrize each node of the network in the same way as (4.7.1). For the ith node:

(4.10.1)

$$\begin{aligned} y_i(t) = &-[A(q^{-1})]_{i1}y_1(t) - \cdots + \{1-[A(q^{-1})]_{ii}\}y_i(t) \\ &+[B(q^{-1})]_{i1}u_1(t) + \cdots + [B(q^{-1})]_{im}u_m(t) + F_i(t) + e_i(t) \end{aligned}$$

where

$$F_i(t) = \sum_{r=1}^{p}\sum_{j=1}^{l} c_j^{ir} f_j^{ir}(t)$$

$$f_j^{ir}(t) = g_j^{ir}[y_r(t-k:t-1), u(t-k:t-1)]$$

$F_i(\cdot)$ is a linear combination of nonlinear basis functions $g_j^{ir}(\cdot)$, depending on the past evolution. Coefficient vector c^i is divided into p groups, each of which represents the regulation from one other node. Within each group c^{ir}, there are l elements corresponding to l basis functions.

Vector c^i is group sparse since some nodes do not control the ith node. In addition, c^i is also element sparse within each group because only a few nonlinear terms match the dynamics of the network. For instance, a group of nonlinear terms (e.g. Hill functions) forms a dictionary to present the potential transcription activity of a transcriptional factor associated to a specific node (gene). The coefficients of this group are 0 if this transcriptional factor does not regulate the target. Besides, only a specific type of Hill functions in this group is

appropriate, which is determined by whether such a transcription is repressive or active.

Following the same framework discussed before, identification of NARX models can be cast as a linear regression problem whose parameter vector is both element and group sparse. Therefore, the developed method in this chapter applies.

4.11 Conclusion

This chapter applies multivariable ARX models to describe sparse networks. Network inference is performed using time series data only. Given limited data and without prior knowledge of the network topology and model complexity, sparsity is imposed to favor sparse topology and model parsimony. The identification problem is formulated as a linear regression. Since the parameter vector is both group (in terms of sparse topology) and element sparse (in terms of reduced system order), the newly proposed method (GESBL) combines group and element SBL to promote both kinds of sparsity. GESBL effectively enforces sparse topology (by using a unified hyperparameter for each group of model parameters). Meanwhile, GESBL also allows a sufficient degree of freedom for parameter estimation (by applying another series of independent hyperparameters for each element within a group). The resulting optimization problem is a blend of reweighted LASSO and group LASSO, which indicates the efficiency of GESBL to impose both kinds of sparsity. This framework is further extended to nonlinear cases.

Overall, the value of this approach is that model parsimony and sparsity of network topology are both imposed at the same time whereas normally only one of them can be achieved by other methods. In addition, the estimation of weighting variables is greatly simplified. While many real-world systems are nonlinear and cannot be well approximated by linear ARX models, GESBL is especially useful to identify NARX models, where group sparsity in terms of sparse topology and element sparsity with regard to the basis selection are equally important.

However, our approach cannot be used to identify other complex linear models (e.g. ARMAX and dynamical structure functions (DSF)) because identification of these parametrized models cannot be formulated as a linear regression. Moreover, stability of these models as *a priori* condition is not reflected by the proposed prior distribution in our framework. Hence, there is no guarantee that estimated models are stable.

Further developments should extend this framework to identify more general models. Additionally, novel prior distributions must be developed to impose system stability. One of the options to solve these problems is kernel methods, which is discussed in the next chapter.

Chapter 5.

Sparse network inference based on kernel methods

Biological networks are complex dynamical systems containing a large number of feedback loops. Accurate mathematical modeling of such networks highly relies on the selected model class. Normally, complex models are more capable of capturing internal dynamics of a network than simple models. Nevertheless, identification of complex models is difficult and, sometimes, requires prior knowledge. NARX models discussed in the last chapter are powerful for interpreting interactions among biological units, however, prior knowledge about internal working of the target network is necessary for their construction. In contrast, establishment of black-box linear models demands no prior knowledge. We have used linear ARX models as a specific type of linear systems to describe a network. Although ARX models have a simple structure, they are not able to represent complex dynamics of most real-world networks. Therefore, we resort to linear state space models that generalize linear systems.

In practice, only a relatively small number of biological units can be measured. Unless there is priori information on unmeasured units, they cannot be modelled and thus are hidden nodes in the network. While most methods to identify nonlinear models require full state measurements, linear models can handle hidden nodes by encoding them as transfer functions. For an ARX model, the information of hidden nodes is implicitly encoded in polynomial terms. Roughly speaking, the number of hidden nodes is a monotonically increasing function of polynomial orders. However, applications of ARX models are limited by their 'model power'. Hence, this chapter applies a general linear state space model to describe biological networks so that all the nodes of a network are explicitly represented by state variables. This chapter first explains how to remove hidden nodes from linear state space models. Then, by expressing a linear model non-parametrically, the problem of model selection is greatly simplified because there is no need to specify the number of hidden nodes and their corresponding biological interpretations.

Biological networks are naturally stable dynamical systems. Hence, it is necessary to model a biological network with a stable mathematical model. Stability analysis of a nonlinear model is difficult and enforcing system stability during its identification is even more challenging. Although it is possible to impose system stability to a parametrized linear model, the resulting identification problem can be very difficult to solve. This chapter presents a method that

enforces system stability in a non-parametric way, so that the inference problem is well-posed and easy to solve.

In this chapter, we introduce dynamical structure functions (DSF) to describe a network. The model is derived from state space expressions whose hidden states are removed from the model. We express DSF models in a non-parametric way, where system properties are characterized by impulse responses. The objective of identification is to estimate these impulse responses. Hence, in contrast to the parameter estimation of one-to-one and GESBL, we consider functional estimation instead. To guarantee system stability, a functional space called reproducing kernel Hilbert space (RKHS) is established using a kernel function, any function within which is a stable impulse response. Consequently, the identification problem is formulated as to find the optimal impulse responses in that functional space. The problem can also be cast from Bayesian perspectives. Empirical Bayes is applied to optimize the hyperparameters of the kernel function that control network topology and internal dynamics. Simulations on ARX models show that this kernel approach is robust to process noise and effectively produces sparse networks. Further tests on dynamical structure functions (DSF) imply its strong ability to infer general linear networks. All the simulation results will be presented in chapter 6 to have a comparison with other methods.

Overall, the advantage of the kernel method is that it guarantees the internal stability of generated networks and avoids problematic model selection. The main drawback is that the proposed optimization problem is highly nonlinear so the method suffers from local optima. In addition, the kernel method does not allow evaluation of inference confidence since the true probabilistic model is approximated by an analytical function.

The main contribution of this chapter is to formulate the system identification problem of DSFs in a non-parametric way and solve the problem using prevalent kernel methods that have been intensively studied in recent years. In addition, the identification problem with measurement noise is discussed.

Section 5.1 provides an overview of kernel methods. Section 5.2 introduces reproducing Hilbert space (RHKS) that is the core of kernel methods. Section 5.3 relates RKHS with Gaussian process so as to build up a bridge between the kernel and Gaussian process regression. Section 5.4 discusses solving regression problems using kernel methods while section 5.5 recasts kernel regression under the Bayesian framework. Section 5.6 introduces feasible kernel functions for RKHS of stable impulse responses. Section 5.7 proposes DSF models to describe networks. Section 5.8 formulates the network reconstruction problem. Section 5.9 discusses Bayesian estimation of impulse responses. Section 5.10 applies empirical Bayes for hyperparameter estimation of kernel functions. Section 5.11 discusses models with measurement noise. Finally, section 5.12 concludes the whole chapter.

5.1 Kernel methods in system identification and machine learning

System identification and machine learning are two deeply studied areas. However, there was little interaction between them until a few ago [118]. For linear system identification, ML and PEM as the mainstream of identification techniques have been devoted to a long history

of theoretical contributions. In the field of machine learning, models are trained to learn the underlying patterns contained in observed data. In terms of diverse demands of applications (e.g. pattern recognition, data classification and regression), various methods have been developed [90], [40]. It is believed that these methods have much to offer to solve system identification problems.

In many applications, the target function to be estimated has special properties. For example, in system identification, the impulse response of a dynamic system is normally stable. Hence, the feasible solution is included in a set of functions that possess the same properties. One can construct a dictionary of candidate functions. The resulting functional space is a finite dimensional vector space. Consequently, functional estimation is cast as a basis selection problem (e.g. [180], [68], [170]). Nevertheless, building a suitable dictionary can be a combinatorial task. Also, since only finite number of basis functions are considered, it is highly possible that the target function cannot be well approximated by their linear combinations. An alternative is to parametrize a candidate function with unknown variables. In this way, a family of functions is characterized by variables in a finite dimensional Euclidean space. For example, empirical Bayes uses a class of distributions, $\hat{p}(w|y,\beta)$ that are dominated by hyperparameter β to approximate the true posterior distribution, $p(w|y)$. It is also a fundamental routine applied by parametric system identification techniques, where a model structure maps a space of model parameters to a set of models. In either way, the target function is characterized by a finite dimensional vector space (e.g. functional space and Euclidean space).

The power of kernel methods lies on a particular infinite dimensional functional space called Reproducing Kernel Hilbert Space (RKHS) which is employed as the feasible domain of functional estimation. Without doubt, RKHS includes much richer forms of functions thus offering a better approximation. More importantly, since there is a one-to-one correspondence between RKHS and kernel functions, the construction of a RKHS is converted to the design of kernel functions. Consequently, a RKHS is the closure of the span of a kernel function [181].

Kernel methods have been merged with the theory of Gaussian process and applied with increasing frequency in system identification community nowadays. They have been used to identify both nonlinear and linear systems. For example, a mixture of Gaussian kernels were used to identify linear parameter-varying (LPV) Box-Jenkins models [114]. The same type of kernel was employed to identify general nonlinear systems including NFI, NARX and NARMAX without specifying the model structure [182]. Kernel methods were also used to identify systems with input uncertainties, including Hammerstein models and cascaded linear systems [106]. The stable spline kernel was developed to identify stable linear SISO continuous-time systems [183]. It was later applied to identify linear discrete-time systems [91], [106], [113], [116], [184].

Kernel-based methods have been used to infer biological networks, representing gene regulation, metabolic process and protein-protein interactions, where a substantial part of the network is known *a priori* [185]–[190]. Connectivity among biological units is indicated by a distance measure of projected nodes in a feature space whereas internal dynamics of the network remain unknown.

In this chapter, we infer biological networks using model-based methods so that network

topology and internal dynamics are both learned from data. Black-box linear state space models are adopted to describe the target network. A kernel-based method is applied to estimate impulse responses of the propose linear model. The stable spline kernel is used to impose stability. Automatic Relevance Determination (ARD) is introduced to promote sparse network topologies.

5.2 Reproducing kernel Hilbert space

5.2.1 Norm, inner product, Banach space and Hilbert space

In general, kernel methods propose an optimization problem, where feasible solutions are constrained in a predefined functional space called RKHS. To introduce RKHS, we first briefly review some basic concepts in functional analysis. A thorough treatment can be found in [191], [192].

A vector space over the field of real numbers is a set of objects (vectors) equipped with operations of addition and scalar multiplication that satisfy certain axioms, for example, commutative and associative laws for addition, and distributive and associative laws for scalar multiplication.

To measure distance between objects of a general set, a function called metric that maps each pair of objects to a non-negative number is defined. In a vector space, the metric is induced by the norm of vectors. The norm satisfies axioms of points separation, positive homogeneity and triangle inequality. A vector space with a norm is called a normed vector space, which induces the concept of convergence of sequences.

A sequence of objects in a vector space is said to be a Cauchy sequence if the distance between any two elements converges to 0. It is known that every convergent sequence is Cauchy but a Cauchy sequence may not converge.

A vector space is said to be complete if every Cauchy sequence converges to a vector of this space. A complete normed space is called a Banach space.

To define geometrical concepts such as angles between vectors, a notation of inner product is introduced. It is a function that maps each pair of vectors to a real number. An inner product is positive definite, symmetric and linear in its first argument. A vector space equipped with an inner product is said to be an inner product space or pre-Hilbert space. One can always define a norm induced by an inner product so a pre-Hilbert space is also a normed space.

A space is said to be a Hilbert space if it is a complete inner product space (i.e. Banach space with an inner product).

A linear operator is a function that maps each vector of a normed space into another normed space and satisfies properties of homogeneity and additivity. If the image of an operator (not necessarily linear) is contained in $\mathbb{R}$, the operator is called a functional. It is known that the following three statements are equivalent: a) a linear operator is continuous at one point, b) a linear operator is continuous at all points, c) a linear operator is bounded.

In a Hilbert space $\mathcal{H}$, all continuous linear functionals L can be expressed in the inner product form $L = \langle \cdot, g \rangle$ for some $g \in \mathcal{H}$ according to the Riesz representation theorem.

5.2.2 Definition of Reproducing Kernel Hilbert Space (RKHS)

There are three key words contained in RKHS: a) reproducing, b) kernel and c) Hilbert.

'Reproducing' originates from a particular property of a RKHS (reproducing property). 'Kernel' implies that a RKHS is associated with kernel functions. 'Hilbert' reveals that a RKHS is a specific type of Hilbert space consisting of functions. We begin with a straightforward definition of RKHS. To distinguish a RKHS from other Hilbert spaces, we first introduce evaluation functional.

Definition 5.2.1 (Evaluation functional) [193]**:** Let $\mathcal{H}$ be a Hilbert space of functions $f: X \to \mathbb{R}$ defined on a non-empty set X. For a fixed $x \in X$, map $\delta_x: \mathcal{H} \to \mathbb{R}$ is a functional that evaluates a function of $\mathcal{H}$ at the point x, $\delta_x(f) = f(x)$.

The evaluation functional of a Hilbert space is not necessarily continuous (bounded). A RKHS is defined to be a functional Hilbert space with a continuous evaluation functional. Surprisingly, with this simple condition, a RKHS is particularly well-behaved compared with other Hilbert spaces. The special property that makes RKHS very useful is called reproducing property.

Definition 5.2.2 (RKHS) [194]**:** A Hilbert space $\mathcal{H}$ of functions $f: X \to \mathbb{R}$ defined on a non-empty set X is said to be a reproducing kernel Hilbert space if its evaluation functional is continuous for any $x \in X$.

5.2.3 Reproducing kernels and kernel functions

As mentioned before, a RKHS is related to a kernel function. Actually, a RHKS is characterized by a reproducing kernel. It will be shown later that a kernel function is equivalent to a reproducing kernel. However, currently, we regard these two as different concepts. We begin with the definition of reproducing kernels.

Definition 5.2.3 (reproducing kernels) [195]**:** Let $\mathcal{H}$ be a Hilbert space of functions $f: X \to \mathbb{R}$ defined on a non-empty set X. A function $k: X \times X \to \mathbb{R}$ is called a reproducing kernel of $\mathcal{H}$, if:
1). $\forall x \in X,\ k(\cdot, x) \in \mathcal{H}$
2). $\forall x \in X,\ \forall f \in \mathcal{H},\ \langle f, k(\cdot, x)\rangle_{\mathcal{H}} = f(x)$ (reproducing property)

The relation between a RKHS and a reproducing kernel is indicated by the following theorem.

Theorem 5.2.1 (Existence and uniqueness of reproducing kernels) [193]**:** A Hilbert space is a RKHS if and only if it has a reproducing kernel. Moreover, this reproducing kernel is unique.

Theorem 5.2.1 implies that each RKHS is associated with a unique reproducing kernel. Hence, we can define a map from a set of RKHS to a set of reproducing kernels. Equivalently, a Hilbert space has a reproducing kernel if and only if its evaluation functional is continuous. It is easy to see that the reproducing kernel of a RKHS is associated to the evaluation functional as $\delta_x(f) = f(x) = \langle f, k(\cdot, x)\rangle_{\mathcal{H}}$, which coincides with the Riesz representation theorem.

In addition to the reproducing property, a reproducing kernel has another important feature that helps unify the concept of reproducing kernels and kernel functions.

Definition 5.2.4 (positive definite functions) [40]**:** A symmetric function $h: X \times X \to \mathbb{R}$ is positive definite if $\forall n \geq 1,\ \forall y \in \mathbb{R}^n,\ \forall x_i \in X, i = 1, \cdots, n$:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j h(x_i, x_j) \geq 0$$

or equivalently

$$C = \begin{bmatrix} h(x_1, x_1) & \cdots & h(x_1, x_n) \\ \vdots & \ddots & \vdots \\ h(x_n, x_1) & \cdots & h(x_n, x_n) \end{bmatrix} \succcurlyeq 0$$

It can be shown that for any Hilbert space $\mathcal{H}$, the inner product can induce a positive definite function as $h(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle_{\mathcal{H}}$ where $\phi: X \to \mathcal{H}$. A simple extension implies that a reproducing kernel is positive definite since by letting $\phi(x) = k(\cdot, x)$, $k(x_1, x_2) = \langle k(\cdot, x_1), k(\cdot, x_2) \rangle_{\mathcal{H}}$.

In contrast to reproducing kernels, a kernel function is defined below:

Definition 5.2.5 (Kernel functions) [193]**:** A function $k: X \times X \to \mathbb{R}$ is said to be a kernel if there exists a Hilbert space $\mathcal{H}$ and a map $\phi: X \to \mathcal{H}$ such that $\forall x_1, x_2 \in X$:

$$k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle_{\mathcal{H}}$$

Function ϕ is called a feature and space $\mathcal{H}$ is a feature space in machine learning [90]. Based on the definition, a reproducing kernel is a kernel function because for RKHS $\mathcal{H}$, $k(x_1, x_2) = \langle k(\cdot, x_1), k(\cdot, x_2) \rangle_{\mathcal{H}}$ where $\phi(x) = k(\cdot, x)$. However, it is not certain whether a kernel function is a reproducing kernel. Another important question is that whether it is possible to construct a Hilbert space using any kernel function and how positive definite functions are involved.

5.2.5 Construction RKHS from kernel functions

To answer the questions above, we begin with constructing a RKHS from a special space called pre-RKHS. We then argue that any positive definite function can be used to establish a pre-RKHS where the function has the reproducing property. A RKHS is then generated by completing the pre-RKHS. Hence, a positive definite function is a reproducing kernel. Finally, we are able to conclude that reproducing kernels, positive definite functions and kernel functions are equivalent.

To construct a RKHS, we first define a pre-RKHS as:

Definition 5.2.6 (pre-RKHS) [193]**:** An inner product space $\mathcal{H}_0$ of functions $f: X \to \mathbb{R}$ is said to be a pre-RKHS if:

1). The evaluation functional δ_x is continuous on $\mathcal{H}_0$.

2). Any Cauchy sequence $\{f_n\}$ in $\mathcal{H}_0$ that converges pointwise to 0 also converges in $\mathcal{H}_0$ norm to 0.

Based on a pre-RKHS, we can construct a RKHS according to the following theorem:

Theorem 5.2.2 (construction of RKHS) [193]**:** Let $\mathcal{H}$ be a set of all functions $f: X \to \mathbb{R}$, for which there exists a Cauchy sequence $\{f_n\}$ in $\mathcal{H}_0$ converging pointwise to f. Then the space $\mathcal{H}$ is a RKHS equipped with the inner product:

$$\langle f, g \rangle_{\mathcal{H}} = \lim_{n \to \infty} \langle f_n, g_n \rangle_{\mathcal{H}_0}$$

where $\{f_n\}$ and $\{g_n\}$ are any Cauchy sequences of $\mathcal{H}_0$ converging pointwise to f and g respectively. The corresponding reproducing kernel is the representer of the evaluation functional (according to Riesz theorem).

Theorem 5.2.2 shows that once a pre-RKHS is constructed, we can use it to deduce a RKHS. The question is how to build a pre-RKHS using a positive definite function and how this function is related to the reproducing kernel of the resulting RKHS in the theorem above. It turns out that a pre-RKHS can be obtained by a finite linear span of functions derived from a positive definite function.

Theorem 5.2.3 (Moore-Aronszajn) [196]**:** Let $k: X \times X \to \mathbb{R}$ be a positive definite function. Let $\mathcal{H}_0 = span\{k(\cdot, x) | x \in X\}$ equipped with the inner product:

$$\langle f, g \rangle_{\mathcal{H}_0} = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(y_j, x_i)$$

where $f = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i)$ and $g = \sum_{j=1}^{n} \beta_i k(\cdot, y_j)$.

Then $\mathcal{H}_0$ is a pre-RKHS. Moreover, k is a reproducing kernel and is associated to a unique RKHS. (This RKHS can be derived from $\mathcal{H}_0$ using the last theorem.)

Summarizing all the discussions above, we have following statements: a) a reproducing kernel is a positive function, b) any positive function is a reproducing kernel, c) any reproducing kernel is a kernel function and d) any kernel function is positive definite. It is then easy to see that reproducing kernels, positive definite functions and kernel functions are exactly the same (see Figure 5.2.1). Hence, we can also construct a RKHS using any kernel function according to the Moore-Aronszajn theorem. Since a kernel function is associated with a unique RKHS, the map from the set of RKHS to the set of kernel functions is bijective.

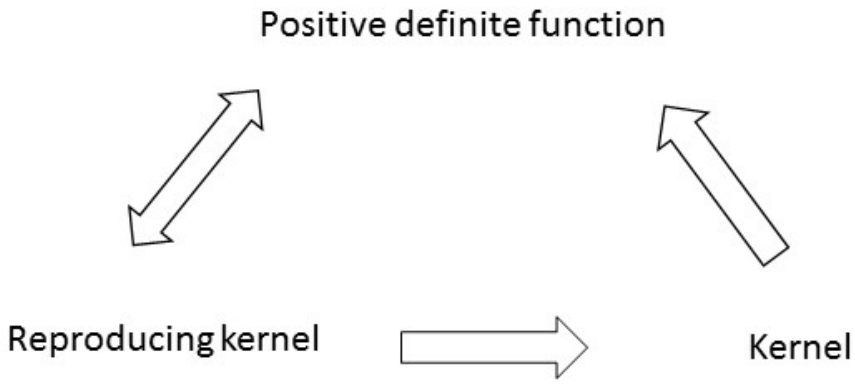


Figure 5.2.1: Relation among reproducing kernels, positive functions and kernels. One concept is implied by another if there is an arrow pointing to it. Any concept can be reached from any other one so these three concepts are equivalent.

Remark 5.2.1: It is now clear that we can use any kernel function to construct a covariance matrix since a kernel is positive definite, which builds a bridge between kernel methods and Gaussian regression. Actually, a RKHS has a statistical interpretation that will be discussed later.

5.2.6 Operations of kernels

One of the most widely used kernels is the Gaussian kernel, $k(x_1, x_2) = exp\,(-\frac{\|x_1-x_2\|_2^2}{2\sigma^2})$. It characterizes a RKHS consisting of smooth functions. There are many other kernels such as polynomial kernels, Fisher kernels and exponential kernels [40]. By using operations of kernels, we are able to construct novel kernels using the existing kernel functions.

Two basic operations are sum and scale of kernels. For any two kernels, k_1 and k_2 defined on $X \times X$, $k_3 = \alpha k_1 + \beta k_2$ is also a valid kernel for $\mathcal{H}_{k_3}$ given $\alpha, \beta \geq 0$. This property inspires multi-kernel regression, where a kernel is represented by a linear combination of multiple kernel functions and Automatic Relevance Determination (ARD) is used to select proper kernels [91]. Furthermore, if k_1 and k_2 are non-negative functions, $\mathcal{H}_{k_3} = \{f_1 + f_2 | f_1 \in \mathcal{H}_{k_1}, f_2 \in \mathcal{H}_{k_2}\}$ is a RHKS with the norm $\|f\|_{\mathcal{H}_{k_3}}^2 = \min_{f=f_1+f_2} \{\|f_1\|_{\mathcal{H}_{k_1}}^2 + \|f_2\|_{\mathcal{H}_{k_2}}^2\}$.

The product of kernels is also a kernel. For any two kernels, $k_1: X \times X \to \mathbb{R}$ and $k_2: Y \times Y \to \mathbb{R}$, $k_3((x_1, y_1), (x_2, y_2)) = k_1(x_1, x_2)k_2(y_1, y_2)$ is a kernel on $X \times Y$.

Other operations include composition, exponentiation and weighting [90].

5.2.7. Mercer representation of RKHS

Every RKHS contains a pre-RKHS consisting of a finite linear span of kernel functions. It was shown that a pre-RKHS is dense in its corresponding RKHS [195]. Although functions in a pre-RKHS have clear expressions, the limit of a pre-RKHS is still ambiguous. It requires no constraints on kernels and metric space X to establish a RKHS. Nevertheless, with additional

assumptions, Mercer's theorems develop a unified expression for all the functions of a RKHS.

To begin with, we assume domain X of a kernel function is a compact metric space and the kernel is continuous (Mercer kernel) [90]. Therefore, the kernel can be used to define a linear map.

Definition 5.2.7 (Integral operator) [115]**:** Let k be a continuous kernel on a compact metric space X. A linear map $T_k: \mathrm{L}_2(X, v) \to \mathrm{C}(X)$ is well-defined by:

$$(T_k f)(x) = \int k(x, y) f(y) dv(y)$$

where $f(y) \in \mathrm{L}_2(X, v)$ is a function of L_2 space endowed with a finite Borel measure v. T_k is a map from L_2 space to space $\mathrm{C}(X)$ of continuous functions. T_k is a self-adjoint, positive and compact operator on L_2 space. (Space of continuous functions is dense in L_2 space.)

Based on the spectral theorem of operators, one can construct an orthonormal basis of L_2 space in terms of the integral operator.

Theorem 5.2.4 (Spectral theorem) [191]**:** There is an at most countable orthonormal basis set $\{e_i\}$ of L_2 space corresponding to positive eigenvalues $\{\lambda_i\}$ of the linear operator T_k that converge to 0 with $\lambda_1 \geq \lambda_2 \geq \cdots$ and $\sum_i \lambda_i < +\infty$ so that:

$$T_k f = \sum_i \lambda_i \langle f, e_i \rangle_{L_2} e_i$$

where $f(y) \in \mathrm{L}_2(X, v)$ and $\{e_i\}$ is a set of eigenvectors of T_k thus continuous functions:

$$\lambda_i e_i = \int k(x, y) e_i(y) dv(y)$$

Mercer's theorem indicates that a kernel function can be represented by the eigenvectors of its integral operator.

Theorem 5.2.5 (Mercer's theorem) [74]**,** [195]**:** Let k be a continuous kernel on a compact metric space X and v a finite Borel measure with support X. Then for $\forall x, y \in X$:

$$k(x, y) = \sum_i \lambda_i e_i(x) e_i(y)$$

where the convergence is uniform and absolute on $X \times X$.

Based on Mercer's theorem, we have another form of description of functions in a RKHS.

Theorem 5.2.6 [193]**,** [119]**:** Let k be a continuous kernel on a compact metric space X. Define:

$$\mathcal{H} = \left\{ f(x) = \sum_i a_i e_i(x) \,|\, \sum_i \frac{a_i}{\sqrt{\lambda_i}} < +\infty \right\}$$

The infinite sum is valid and convergence is defined in $\mathbb{R}$. Moreover, $\mathcal{H} = \mathcal{H}_k$ with the inner product:

$$\langle \sum_i a_i e_i(x), \sum_i b_i e_i(x) \rangle = \sum_i \frac{a_i b_i}{\lambda_i}$$

Additionally, the family $\{\sqrt{\lambda_i} e_i\}$ forms an orthonormal basis of $\mathcal{H}$.

It was further shown that the convergence in Theorem 5.2.6 is uniform and absolute [195]. Since e_is are continuous functions, $f(x)$ is also continuous. In addition, by taking the square integral of $f(x)$ and using Cauchy-Schwartz inequality, the integral is finite meaning $f(x) \in \mathrm{L}_2(X, v)$. To conclude, we have the following theorem:

Theorem 5.2.7 [194]**,** [195]**:** $\mathcal{H}_k \subset \mathrm{C}(X)$. $\mathcal{H}_k \subset \mathrm{L}_2(X, v)$.

Since a RKHS is characterized by a unique kernel, the property of functions in a RKHS also depends on the kernel. A Mercer kernel defined on a compact metric space is square integrable. Functions of a RKHS endowed with a Mercer kernel also share the same property according to Theorem 5.2.7. This implies that constructing a RKHS composed of functions with desired properties is equivalent to designing kernel functions that hold the same properties.

Mercer representation can be further extended to non-compact metric spaces, where similar theorems apply [181]. Mercer theorem has been used to analyze the RKHS of impulse responses of stable linear continuous time systems [183].

5.3 RKHS and stochastic process

A stochastic process $X(t)$ is a family of random variables indexed by t in an index set, T. Since it is a random variable at each time instance, $X(t)$ is partially characterized by mean $m(t)$, second moment $r(t,s)$ and covariance function $k(t,s)$:

(5.3.1)

$$\begin{aligned} m(t) &= E\{X(t)\} \\ r(t,s) &= E\{X(t)X(s)\} \\ k(t,s) &= E\{[X(t) - m(t)][X(s) - m(s)]\} \\ &= r(t,s) - m(t)m(s) \end{aligned}$$

As covariance matrix K with $[K]_{ij} = k(x_i, x_j)$ is positive semidefinite, covariance function $k(t,s)$ is a positive definite function based on Definition 5.2.4. Hence, covariance functions and kernels are equivalent. We can use a kernel function to define a stochastic process.

A stochastic process can be represented by a Hilbert space according to the definition below.

Definition 5.3.1 (representation of stochastic processes) [106]**:** A family of vectors $f_t : t \in T$ in a Hilbert space $\mathcal{H}$ is a representation of the second order stochastic process if

$$\langle f_t, f_s \rangle_{\mathcal{H}} = r(t,s)$$

According to Definition 5.3.1, it is easy to link a RKHS to a stochastic process.

Theorem 5.3.1 [197]**:** Let $X(t)$ be a zero mean stochastic process with $r(t,s) = E\{X(t)X(s)\}$ and $\mathcal{H}_k$ be the RKHS with respect to the kernel $r(t,s)$. Then the family of functions $\{r(\cdot,t) \in \mathcal{H}_k\}$ is a representation of the stochastic process $X(t)$.

Theorem 5.3.1 only associates a family of functions of a RKHS to a stochastic process. In fact, we can link the entire space of a RKHS to a Hilbert space constructed by a stochastic process.

It is known that space L_2 of real-valued random variables x with finite second moments $E(x^2) < +\infty$ is a Hilbert space endowed with the inner product $\langle x,y\rangle_{L_2} = E(xy)$. Consider a space l as a finite linear span of a stochastic process $X(t)$: $l = \{y|y = \sum_{i=1}^{n} \alpha_i X(t_i), n = 1,2,\cdots\}$. l is a subspace of L_2 and its closure L is also a Hilbert space with the inner product $\langle x,y\rangle_L = E(xy)$. Each element x of L can be represented as the limit of a sequence of random variables $\{x_n \in l\}$ of space l: $x = \lim_{n\to\infty} x_n$. As a result, $\|x\|_L = \lim_{n\to\infty} \|x_n\|_L$ and $\langle x,y\rangle_L = \lim_{n\to\infty}\lim_{m\to\infty} < x_n, y_m >_L$. On the other hand, pre-RKHS $\mathcal{H}_0$ consisting of a linear span of the kernel function $\{r(\cdot,t)\}$ is dense in RKHS $\mathcal{H}_k$. There is a one-to-one correspondence between $\mathcal{H}_0$ and l defined by a map U, which preserves the inner product:

(5.3.2)

$$U: \sum_{k=1}^{n} a_k r(\cdot,t_k) \to \sum_{k=1}^{n} a_k X(t_k)$$
$$< f_1, f_2 >_{\mathcal{H}_0} =< Uf_1, Uf_2 >_l = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j r(t_i,t_j)$$

In addition, to consider the entire RKHS, for any $f = \lim_{n\to\infty} f_n$ where $f_n \in \mathcal{H}_0$, $Uf = \lim_{n\to\infty} Uf_n \in L$ is well-defined. Hence, it is easy to prove that U is a linear bijective map from $\mathcal{H}_k$ to L. Furthermore, the inner product under this map is preserved: $< f,g >_{\mathcal{H}_k} = \lim_{n\to\infty}\lim_{m\to\infty} < f_n, g_m >_{\mathcal{H}_k} = \lim_{n\to\infty}\lim_{m\to\infty} < Uf_n, Ug_m >_L =< Uf, Ug >_L$. To conclude, Hilbert space L induced by stochastic process $X(t)$ with covariance function $r(t,s)$ is isometrically isomorphic to the RKHS endowed with the same kernel function $r(t,s)$.

Theorem 5.3.2 [119]**,** [106]**:** The closed linear span $L = \overline{span\{X(t)\}}$ of the zero mean stochastic process $X(t)$ with second moment $r(t,s)$ is isometrically isomorphic to the RKHS with kernel $r(t,s)$.

This one-to-one correspondence implies that a problem posed in a RKHS can be recast in a space constructed by stochastic processes, which builds a foundation to discuss kernel methods under the Bayesian framework.

A Gaussian process is a specific type of stochastic processes. It is completely characterized

by mean and second moment. Gaussian processes are mathematically equivalent to many models such as Bayesian linear models, neural networks and spline models. Hence, they provide a probabilistic approaches to learning in kernel machines [74].

Definition 5.3.2 [74]**:** A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

With a kernel function, we can completely define a zero mean Gaussian process whose covariance function is that kernel. Therefore, Gaussian processes are widely used under the Bayesian framework, which inherently adopt the spirit of kernel methods.

5.4 Interpolation based on kernel methods

Consider a generative model:

$$y(t) = (Tf)(t) + \varepsilon(t) \tag{5.4.1}$$

where $t \in D$, D is the domain for functions (e.g. $\mathbb{R}_+$ and $\mathbb{Z}_+$), ε is i.i.d. Gaussian noise with variance σ^2, f is a function in RKHS $\mathcal{H}_k$ endowed with kernel $k(t,s)$ and T is a linear operator with domain $\mathcal{H}_k$. For example, T can be an operator $(Tf)(t) = \int_0^t f(t-\tau)u(\tau)d\tau$ which can be interpreted as the convolution of input $u(t)$ with impulse response $f(t)$ of a dynamic system. Note that this operator can be directly extended to the discrete-time case. If T is simply an inclusion map: $(Tf)(t) = f(t)$, the model becomes $y(t) = f(t) + \varepsilon(t)$, which is a basic regression model.

Given the observed data of $y(t)$, the objective is to estimate function f. In practice, only finite data points are available so the collected data are presented as a sequence $\{y(t_i), i = 1,2,\cdots,n\}$. The model can be rewritten as:

$$y_i = T_i f + \varepsilon_i \tag{5.4.2}$$

where $y_i = y(t_i)$, δ_{t_i} is an evaluation functional $\delta_{t_i} g = g(t_i)$ and $T_i = \delta_{t_i} \circ T$ is a linear functional. If T is a convolution operator, then $T_i f = \int_0^{t_i} f(t_i - \tau)u(\tau)d\tau$. For an inclusion map T, $T_i = \delta_{t_i} \circ T = \delta_{t_i}$ and the model becomes $y_i = f_i + \varepsilon_i$ where $f_i = f(t_i)$.

If RKHS $\mathcal{H}_k$ is decomposed as the direct sum of two orthogonal vector spaces: $\mathcal{H}_k = \mathcal{H}_0 \oplus \mathcal{H}_1$ where the dimension of $\mathcal{H}_0$ is m. It is easy to show that Hilbert space $\mathcal{H}_0$ is a RKHS with kernel $k_0(t,s) = \sum_{i=1}^m \phi_i(t)\phi_i(s)$ where $\{\phi_i, i = 1,2,\cdots m\}$ is the orthonormal basis of $\mathcal{H}_0$. In addition, $\mathcal{H}_1$ is also a Hilbert space because every Cauchy sequence converges to a vector within $\mathcal{H}_1$. Moreover, $\mathcal{H}_1$ is a RKHS since its evaluation functional is bounded. (Evaluation functional of the whole space, $\mathcal{H}_k$ is bounded.) Let $k_1(t,s) = k(t,s) - k_0(t,s)$. Then $k_1(\cdot,s)$ is a function of $\mathcal{H}_1$ for $\forall s \in D$ since it is orthogonal to $\mathcal{H}_0$. Furthermore, $< f, k_1(\cdot,s) >_{\mathcal{H}_k} = f(s)$ for $\forall f \in \mathcal{H}_1$. Consequently, $k_1(t,s)$ is the kernel for $\mathcal{H}_1$ according to Theorem 5.2.3.

Assuming linear functional T_i is bounded, an estimate $\hat{f}$ of f is the solution of the

following regularized optimization problem [119]:

(5.4.3)

$$\hat{f} = \arg\min_{f \in \mathcal{H}_k} \frac{1}{n} \sum_{i=1}^{n} (y_i - T_i f)^2 + \lambda \|P_1 f\|_{\mathcal{H}_k}^2$$

where P_1 is the orthogonal projection of f onto $\mathcal{H}_1$ in $\mathcal{H}_k$. The decomposition of $\mathcal{H}_k$ depends on the requirement of interpolation. For example, in spline smoothing problems, $\mathcal{H}_k$ can be a Sobolev-Hilbert space so $\mathcal{H}_1$ is a space of functions whose first $m-1$ derivatives $\{D^v f, v = 1,2,\cdots,m-1\}$ are absolute continuous satisfying $(D^v f)(0) = 0$ and the mth derivative is in L_2 space, and $\mathcal{H}_0$ is the m dimensional space of polynomials. In this case, $\|P_1 f\|_{\mathcal{H}_k}$ penalizes the power of the projection of f in $\mathcal{H}_1$ as $\|P_1 f\|_{\mathcal{H}_k}^2 = \int_0^1 (D^m P_1 f)^2(u) du$ [119]. In system identification, $\mathcal{H}_0$ can be a space of impulse responses possessing dominant poles of the target system and $\mathcal{H}_1$ is a RKHS of stable impulse responses [183].

In contrast to the normal regularized optimization problems in section 4.2.1, the feasible set of solutions to (5.4.3) is an infinite dimensional functional space rather than a finite dimensional Euclidean space. At first glance, solving such a problem is highly complex. Nevertheless, it turns out that the solution can be expressed analytically by the merit of the reproducing property.

Theorem 5.4.1 [119]**:** Let $\{\phi_i, i = 1,2,\cdots m\}$ be orthonormal basis of $\mathcal{H}_0$ and define a $n \times m$ matrix N of full column rank as $[N]_{ij} = T_i \phi_j$. Then the solution $\hat{f}$ to (5.4.3) is given by

$$\hat{f} = \sum_{i=1}^{m} a_i \phi_i + \sum_{j=1}^{n} b_j \xi_j$$

where $\xi_j = P_1 \eta_j$, η_j is the representer of the bounded linear functional T_j according to the Riesz representation theorem so that $< \eta_j, f >_{\mathcal{H}_k} = T_j f, f \in \mathcal{H}_k$ and

$$y = (y_1, y_2, \cdots, y_n)'$$
$$a = (a_1, a_2, \cdots, a_m)' = (N' M^{-1} N)^{-1} N' M^{-1} y$$
$$b = (b_1, b_2, \cdots, b_m)' = M^{-1}[I - N(NM^{-1}N)^{-1}N'M^{-1}]y$$
$$M = \Sigma + n\lambda I$$
$$[\Sigma]_{ij} = < \xi_i, \xi_j >$$

Theorem 5.4.1 generalizes a special case where the decomposition of $\mathcal{H}_k$ is not required and $T_i = \delta_{t_i}$ is the evaluation functional. As a result, problem (5.4.3) becomes:

(5.4.4)

$$f = \arg\min_{f \in \mathcal{H}_k} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_i)^2 + \lambda \|f\|_{\mathcal{H}_k}$$

Hence, Theorem 5.4.1 is simplified to the representation theorem.

Theorem 5.4.2 (representation theorem) [118]**:** Let $k(\cdot,\cdot)$ be the kernel corresponding to RKHS $\mathcal{H}_k$. Then the solution $\hat{f}$ to (5.4.4) is given by

$$\hat{f}(\cdot) = \sum_{i=1}^{n} a_i k(\cdot, t_i)$$

where

$$a = (a_1, a_2, \cdots, a_n)' = (n\lambda I + K)^{-1} y$$
$$y = (y_1, y_2, \cdots, y_n)'$$
$$[K]_{ij} = k(t_i, t_j)$$

As linear dynamic systems are characterized by impulse responses and system outputs are the convolution (linear operator) of impulse responses with inputs, the framework discussed above can be applied to identify non-parametrized linear models (e.g. [183], [113], [198]). In addition, this framework can also be used to identify nonlinear systems (e.g. [114], [182]).

5.5 Bayesian formulation of kernel methods

Theorem 5.3.2 shows that a RKHS is equivalent to a space L of random variables as the closed linear span of a stochastic process in the sense of isometric isomorphism. This relation leads to the duality between Bayesian approaches and kernel methods to solve the same interpolation problem (5.4.1).

The generative model in the functional setting is given by $y_i = T_i f + \varepsilon_i$. We consider reformulating this model under the Bayesian framework. To begin with, we define the counterpart of linear functional T_i on space L spanned by a Gaussian process.

Let T be a bounded linear functional on RKHS $\mathcal{H}_k$ with kernel $k(t,s)$. Then, there exists a representer, $\eta(s) = Tk(s,\cdot)$ for T in $\mathcal{H}_k$ such that $Tf = <\eta, f>_{\mathcal{H}_k}$ for $\forall f \in \mathcal{H}_k$ [199]. Due to isometric isomorphism between $\mathcal{H}_k$ and L that is the closed linear span of zero mean Gaussian process $X(t)$ with covariance function $k(t,s)$, we can define a counterpart, $\tilde{T}$ for T as a linear functional on L with its representer as $z = U\eta \in L$ where U is the bijective linear map from $\mathcal{H}_k$ to L that preserves the inner product (i.e. $< f, g >_{\mathcal{H}_k} = < Uf, Ug >_L$). Hence, $\tilde{T}x = < z, x >_L$ for $\forall x \in L$. As a result, $Tf = < \eta, f >_{\mathcal{H}_k} = < U\eta, Uf >_L = < z, Uf >_L = \tilde{T}(Uf)$. For example, convolution functional T_t on $\mathcal{H}_k$ ($T_t f = \int_0^t f(t-\tau)u(\tau)d\tau$) corresponds to a linear functional, $\tilde{T}_t$ on L with $z = \int_0^t X(t-\tau)u(\tau)d\tau$ as its representer , where the integral is defined in quadratic mean [119] (i.e. expectation of the square of random variables).

Definition 5.5.1: Let $X(t)$ be a zero mean Gaussian process with covariance function $k(t,s)$, L be a Hilbert space that is the closed linear span of $X(t)$ and T be a bounded linear functional on RKHS $\mathcal{H}_k$ with kernel function $k(t,s)$. TX is defined to be a random variable of space L and equates to the representer of the linear functional $\tilde{T}$ on L which is the counterpart of T on $\mathcal{H}_k$.

Now, we are ready to propose a Bayesian probabilistic model as the duality of (5.4.2). For $\mathcal{H}_k = \mathcal{H}_0 \oplus \mathcal{H}_1$ with $k(t,s)$, $k_0(t,s)$ and $k_1(t,s)$ to be their corresponding kernels, we have [119]:

(5.5.1)

$$y_i = T_i F + \varepsilon_i$$
$$F(t) = \sum_{i=1}^{m} \theta_i \phi_i(t) + X(t)$$

where $\theta = (\theta_1, \theta_2, \cdots, \theta_m)'$ is fix but unknown, $\varepsilon_i \sim \mathcal{N}(\varepsilon_i | 0, \sigma^2)$, $\{\phi_i\}$ is a set of orthonormal basis of $\mathcal{H}_0$ and $X(t)$ is a zero mean Gaussian process with covariance function $k_1(t,s)$. With a little abuse of notation, T_i denotes both a bounded linear functional on $\mathcal{H}_k$ and its counterpart on $L^1 = \overline{span\{X(t)\}}$. (Since L^1 is isometrically isomorphic with $\mathcal{H}_1$, T_i (originally defined on $\mathcal{H}_k$) is well-defined on L^1. So $T_i X$ is a random variable that is the representer of T_i on L^1.)

Given observations $\{y(t_i), i = 1,2,\cdots,n\}$, instead of estimating unknown variable θ, Bayesian methods focus on reducing generalization errors of the model. Assuming T_0 is another bounded linear functional on $\mathcal{H}_k$, the goal is to estimate $T_0 F$.

Let $\widehat{T_0 F}$ be the linear estimator of $T_0 F$ that has the minimum variance and is unbiased given θ. The problem is formulated as [119]:

(5.5.2)

$$\min_a E\{\left(\widehat{T_0 F} - T_0 F\right)^2\}$$

Subject to

$$\widehat{T_0 F} = \sum_{i=1}^{n} a_i y_i$$
$$E\left(\widehat{T_0 F} - T_0 F \middle| \theta\right) = 0$$

It turns out that the solution to (5.5.2) is linked to the problem (5.4.3). The unknown parameter θ is implicitly estimated under the Bayesian framework.

Theorem 5.5.1 [119]**:** $\widehat{T_0 F} = T_0 f$ is the solution to (5.5.2)
where f is the solution to the following optimization problem:

$$\arg\min_{f \in \mathcal{H}_k} \frac{1}{n} \sum_{i=1}^{n} (y_i - T_i f_i)^2 + \lambda \|P_1 f\|_{\mathcal{H}_k}$$

with $= \frac{\sigma^2}{n}$.

Theorem 5.5.1 implies that under the context of system identification, we can regard impulse responses of a system as a Gaussian process and estimate impulse responses under the Bayesian framework. The solution of Bayesian estimation is exactly the same with that of

kernel methods. Assuming $T_0 = \delta_{\bar{t}}$ and T_i is a convolution functional, we formulate system identification problems under the functional framework and Bayesian perspectives. For the simplicity of demonstration, we consider a SISO continuous time system. The problem can be solved using Theorem 5.5.1 and 5.4.1.

Objective: Estimate impulse response $f(t)$ of a continuous SISO system
Functional setting:

$$y(t) = \int_0^t f(t-\tau)u(\tau)d\tau + \varepsilon(t)$$

where $f \in \mathcal{H}_k$ with the kernel $k(t,s)$.
Bayesian model:

$$y(t) = \int_0^t F(t-\tau)u(\tau)d\tau + \varepsilon(t)$$

where $F(t)$ is a zero mean Gaussian process with covariance function $k(t,s)$ and the integral is defined in the sense of quadratic mean.

For discrete time systems, the integrals above are replaced by infinite sum. In the Bayesian model, infinite sum is also defined in the sense of quadratic mean.

In practice, kernel functions are usually parametrized by unknown hyperparameters that take charge of the property of kernel functions. Estimating these hyperparameters is important as they determine how the estimated model interprets data. For kernel methods, one common approach is cross-validation, which causes information loss and increases computational burden. By casting kernel methods under the Bayesian framework, Bayesian techniques can be used to estimate hyperparameters. For example, the empirical Bayes method can optimize hyperparameters via type II maximization so that the RKHS of the resulting kernel better characterizes the target function [74]. In what follows, we will formulate network inference problems under the Bayesian framework.

5.6 Kernel functions for impulse responses

As discussed before, we can treat impulse responses of a dynamical system as a zero mean Gaussian process with covariance function $k(t,s)$ (kernel function), set up a probabilistic model and then formulate the identification problem under the Bayesian framework. The only question is what kernels should be used or how to construct a RKHS for impulse responses. The property of functions in a RKHS is determined by its kernel function, which encourages the study of kernel functions to impose the desired dynamical properties of impulse responses.

5.6.1 Non-parametric LTI systems

So far, we have only proposed parametric dynamic models. Under the framework of kernel methods, models are postulated in a non-parametric way, so that the problem of model selection is simplified [118]. Without loss of generality, we consider SISO systems. To handle both continuous time and discrete time systems, we use an abstract time set $\mathcal{O}$ ($\mathcal{O} = \mathbb{R}\ or\ \mathbb{Z}$).

An LTI SISO system is characterized by impulse response $f: \mathcal{O} \to \mathbb{R}$. For any input $u: \mathcal{O} \to$

$\mathbb{R}$, the output of the system $y: \mathcal{O} \to \mathbb{R}$ is generated by the convolution of u and f:

$$y(t) = \int_{\mathcal{O}} f(t-\tau)u(\tau)d\tau \tag{5.6.1}$$

where the convolution can be interpreted as an integral or an infinite sum depending on the nature of time set $\mathcal{O}$.

Instead of estimating parameters of a parametrized model, the identification problem here becomes to estimate the impulse response function, f given noisy measurements.

5.6.2 Enforcing system properties using kernels

In practice, dynamical systems usually have some basic properties. The goal is to construct kernels so that the corresponding RKHS consists of impulse responses that satisfy the desired properties.

One of the most important properties of a real-world system is causality. That means the output signal of a system at a certain time, t_0 does not depend on values of the input in the future ($t > t_0$). For an LTI system, it is equivalent to claiming that values of impulse responses for negative time instances are zero: $f(t) = 0$ for $\forall t < 0$. The following theorem characterizes kernel functions whose corresponding RKHSs consist of causal impulse responses.

Theorem 5.6.1 (causal systems) [199]**:** A RKHS contains only causal impulse responses if and only if the kernel satisfies

$$k(t,s) = H(t)H(s)\tilde{k}(t,s)$$

where $H(t)$ is the Heaviside step function and $\tilde{k}(t,s)$ is a kernel defined for non-negative time instances:

$$H(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

Another important feature of dynamical systems is stability. In most cases, systems to be identified are stable in the sense of BIBO (bounded input bounded output) (i.e. $\|y\|_\infty < +\infty$ if $\|u\|_\infty < +\infty$ where $\|\cdot\|_\infty$ denotes infinite norm). For LTI systems, BIBO is equivalent to absolute integrability of impulse responses (i.e. $\int_{\mathcal{O}} |f(t)|dt < +\infty$). Therefore, a RKHS consisting of stable impulse responses must be a subspace of $L_1(\mathcal{O})$ space. The following theorem provides a necessary and sufficient condition on kernel functions to guarantee system stability.

Theorem 5.6.2 (stable systems) [199]**,** [118]**:** A RKHS is a subspace of $L_1(\mathcal{O})$ space if and only if

$$\int_{\mathcal{O}} \left| \int_{\mathcal{O}} k(t,s)f(t)dt \right| ds < +\infty \text{ for } \forall f \in L_\infty(\mathcal{O})$$

In particular, if $k(t,s) \in L_1(\mathcal{O} \times \mathcal{O})$, then the corresponding RKHS is a subspace of $L_1(\mathcal{O})$.

The last property to be considered is time delay. For causal LTI systems, time delay D is

defined as $D = \inf\{t \in \mathcal{O}: f(t) \neq 0\}$ which is non-negative. If a system has time delay D, output $y(t_0)$ is independent on the input for $\forall t > t_0 - D$. By shifting a kernel function, we have a RKHS composed of time-delayed impulse responses.

Theorem 5.6.3 (time-delayed systems) [199]**:** The RKHS consists of impulse responses with time delay D if and only if

$$k_D(t,s) = k(t-D, s-D)$$

with $k(t,s)$ being a kernel for causal systems.

Theorems above demonstrate conditions on kernel functions that must be met to characterize dynamics of most practical linear systems. They can be used to check whether a proposed kernel function is valid to represent impulse responses.

5.6.3 Stable spline kernel for discrete time systems

The second order stable spline kernel has been used to characterize Gaussian processes that encode BIBO property of impulse responses for continuous time systems [118], [183], [198]:

(5.6.2)

$$k(t,s;\beta) = \frac{e^{-\beta(t+s)}e^{-\beta\max(t,s)}}{2} - \frac{e^{-3\beta\max(t,s)}}{6}$$

where $k(t,s;\beta)$ is controlled by hyperparameter $\beta > 0$ which determines the decaying rate of impulse responses [184]. It has been shown that the realizations of such Gaussian processes are continuous time impulse responses of BIBO stable systems with probability one [183]. Therefore, the stable spline kernel has been used to identify non-parametric continuous time systems under the Bayesian framework.

Remark 5.6.1: Other kernel functions such as Diagonal/Correlated kernel (DC) and Tuned/Correlated kernel (TC) are also widely used in system identification community [112]. Although they have been shown very effective [116], these kernels are mainly verified in problems that focus on estimating input-output relationships of dynamical systems. Hence, their ability to tackle network inference problems is questionable. The stable spline kernel has been applied to infer networks described by Granger causality and achieved great success [113]. Therefore, we adopt this kernel in our framework. Monte Carlo simulations also indicate that the stable spline kernel outperforms other kernel functions.

In what follows, we will use the discrete version of the stable spline kernel to represent impulse responses of discrete time systems ($k: \mathcal{O} \times \mathcal{O} \to \mathbb{R}, \mathcal{O} = \{0,1,2,3,\dots\}$). Since impulse responses of stable continuous time systems decay exponentially, the sampled version of a stable continuous time impulse response is also stable in the sense of discrete time systems. Hence, the realizations of Gaussian processes with the discrete stable spline kernel are stable discrete time impulse responses with probability one. As a result, this kernel can be used in the Bayesian formulation of kernel methods (section 5.5). Under the functional framework of kernel methods (section 5.4), we need to check whether the resulting RKHS of the discrete stable spline kernel is a subspace of $L_1(\mathcal{O})$. In other words, we have to prove $k(t,s) \in L_1(\mathcal{O} \times \mathcal{O})$ according to Theorem 5.6.2.

Theorem 5.6.4: The discrete stable spline kernel $k(t,s)$ is a function in $L_1(\mathcal{O}\times\mathcal{O})$.
Proof: First, note that the stable spline kernel is non-negative. Without loss of generality, we only need to check the situation where $t \geq s$ as the kernel is symmetric with respect to $t = s$. For $t \geq s$,

$$\begin{aligned} k(t,s;\beta) &= \frac{e^{-\beta(t+s)}e^{-\beta t}}{2} - \frac{e^{-3\beta \mathrm{t}}}{6} \\ &\geq \frac{e^{-2\beta t}e^{-\beta t}}{2} - \frac{e^{-3\beta \mathrm{t}}}{6} \\ &= \frac{e^{-3\beta t}}{3} \\ &> 0 \end{aligned}$$

Now, we need to prove $\sum_{t=0}^{\infty}\sum_{s=0}^{\infty}|k(t,s;\beta)| < +\infty$. According to the theorem of double series, we only need to prove $\sum_{t=0}^{\infty}(\sum_{s=0}^{\infty}|k(t,s;\beta)|) = \sum_{t=0}^{\infty}(\sum_{s=0}^{\infty}k(t,s;\beta)) < +\infty$.

To begin with, consider a function $\hat{k}(t,s;\beta) = \frac{e^{-\beta\alpha(t+s)}}{2}$ where $\alpha > 0$. It is easy to see that:

$$\sum_{t=0}^{\infty}\left(\sum_{s=0}^{\infty}\frac{e^{-\beta\alpha(t+s)}}{2}\right) = \frac{1}{2}\left(\frac{1}{1-e^{-\beta\alpha}}\right)^2 < +\infty \text{ for } \forall\alpha > 0,\ \beta > 0$$

If $\exists\alpha$ such that $k(t,s;\beta) \leq \hat{k}(t,s;\beta)$ for $\forall t,s \geq 0$, then the proof is done. To see this, note that:

$$\begin{aligned} &\hat{k}(t,s;\beta) - k(t,s;\beta) \geq 0 \\ \Leftrightarrow\ &3e^{-\beta\alpha(t+s)} - 3e^{-\beta(t+s)}e^{-\beta\max(t,s)} + e^{-3\beta\max(t,s)} \geq 0 \end{aligned}$$

The above inequality is satisfied if

$$\begin{aligned} &3e^{-\beta\alpha(t+s)} - 3e^{-\beta(t+s)} \geq 0 \\ \Leftrightarrow\ &e^{-\beta\alpha(t+s)} \geq e^{-\beta(t+s)} \\ \Leftrightarrow\ &\alpha \leq 1 \end{aligned}$$

As a result, for $\forall\alpha \in (0,1]$, $k(t,s;\beta) \leq \hat{k}(t,s;\beta)$ for $\forall t,s \geq 0$, $\beta > 0$. Hence, $\sum_{i=0}^{\infty}\left(\sum_{j=0}^{\infty}k(t,s;\beta)\right) \leq \sum_{i=0}^{\infty}\left(\sum_{j=0}^{\infty}\hat{k}(t,s;\beta)\right) < +\infty$.

Theorem 5.6.4 indicates that we can use the discrete stable spline kernel to construct a RKHS for causal and stable discrete time impulse responses.

5.7 Sparse linear networks described by DSFs

Since ARX models applied in the last chapter are not suitable to describe networks with complex dynamics, we adopt state space models which are the most general expressions for LTI systems. The state variables represent nodes of a network. A network is driven by both external inputs and process noise. For the simplicity of demonstration, we assume there is no measurement noise in the model:

(5.7.1)

$$\begin{aligned} x(t+1) &= Ax(t) + B_u u(t) + B_e e(t) \\ y(t) &= Cx(t) \end{aligned}$$

where $x \in \mathbb{R}^n$ are states of the system, $u \in \mathbb{R}^m$ denote the inputs, $y \in \mathbb{R}^p$ represent the measurements of states and $e \in \mathbb{R}^q$ are i.i.d. Gaussian noise with zero mean and covariance matrix P_e. Without loss of generality, P_e is assumed to be diagonal. If the covariance matrix is full, one can decompose the matrix as $P_e = R\Sigma R'$ using singular value decomposition (SVD) [171]. Then, B_e can be updated as $B_e^{new} = B_e^{old} R$ and the covariance matrix can be replaced by the diagonal matrix, $\Sigma^{\frac{1}{2}}$. $A \in \mathbb{R}^{n\times n}$, $B_u \in \mathbb{R}^{n\times m}$, $B_e \in \mathbb{R}^{n\times q}$ and $C \in \mathbb{R}^{p\times n}$ are system matrices.

In practice, full state measurements are usually unavailable. For example, in biology, only a small number of biological units (e.g. concentrations of mRNAs) can be measured while the others (e.g. proteins) are unobservable. We treat manifest states (measurable nodes) as outputs and unmeasurable nodes as hidden states. Without loss of generality, we assume the first $p < n$ states are measurable so that C can be written as $C = [I \quad \mathbf{0}]$.

In what follows, we would like to model a network composed of both measured and unmeasured nodes. Moreover, we assume the number of hidden states and their biological interpretations are unknown. To avoid inferring these hidden states, we have to remove them from the model. Dynamical structural function (DSF) is an efficient model that encodes information of hidden states via transfer functions [200], [84]. To begin with, we partition the states into manifest and hidden states:

(5.7.2)

$$\begin{bmatrix} y(t+1) \\ h(t+1) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} y(t) \\ h(t) \end{bmatrix} + \begin{bmatrix} B_u^1 \\ B_u^2 \end{bmatrix} u(t) + \begin{bmatrix} B_e^1 \\ B_e^2 \end{bmatrix} e(t)$$

where $h \in \mathbb{R}^{n-p}$ are hidden states.

A DSF is then built by eliminating the hidden states [84], [85]:

(5.7.3)

$$Y = QY + PU + HE$$

where q denotes the time shift operator so that $y(t+1) = qy(t)$ and:

$$Q = (qI - D)^{-1}(W - D)$$
$$P = (qI - D)^{-1}V_u$$
$$H = (qI - D)^{-1}V_e$$

with

$$W = A_{11} + A_{12}(qI - A_{22})^{-1}A_{21}$$
$$V_u = A_{12}(qI - A_{22})^{-1}B_u^2 + B_u^1$$
$$V_e = A_{12}(qI - A_{22})^{-1}B_e^2 + B_e^1$$
$$D = diag\{W_{11}, W_{22}, \dots, W_{pp}\}$$

Q is a matrix of transfer functions (transfer matrix) indicating the connectivity among manifest nodes. The diagonal elements of Q are zero. Similarly, P and H are transfer matrices relating inputs and noise to nodes, respectively. Note that elements of Q, P and H are strictly proper transfer functions indicating the network is a causal system. The topology of the network is reflected by the zero structure of Q and P matrices. If $[Q]_{ij}$ or $[P]_{ij}$ is non-zero, there is an edge from the jth node or input to the ith node. In addition, dynamics of edges are described by the transfer functions (entries) of Q and P.

From a DSF, we can deduce the input-output map of the system:

(5.7.4)

$$Y = G_u U + G_e E$$

where

$$G_u = (I - Q)^{-1} P$$
$$G_e = (I - Q)^{-1} H$$

With infinite source of data, we can accurately recover the input-output map of a system using identification methods like PEM. However, an input-output map can correspond to many DSF models with different topologies. To ensure the inference problem is well-posed, we first define the reconstruction of DSF from the input-output map.

Definition 5.7.1 *(Reconstruction of DSF):*
Given the input-output map $G = [G_u \quad G_e]$, the DSF can be reconstructed if there exists a unique $[Q \quad P \quad H]$ which satisfies (5.7.4).

Unfortunately, based on the definition, a general network may not be identifiable unless partial structure of the network is known. The following theorem is a simple extension of the work in [84], [200].

Theorem 5.7.1 *(Reconstruction with partial structure):*
Given a $p \times (m + q)$ transfer matrix $G = [G_u \quad G_e]$, the DSF can be reconstructed if and only if $p - 1$ elements in each column of $[Q \quad P \quad H]'$ are known that uniquely specify the component of (Q, P, H) in the null space of $[G' \quad I]$.

A sufficient condition for reconstructing DSF is that P or H is diagonal so that $p - 1$ elements in each column of $[Q \quad P \quad H]'$ are known to be 0.

5.8 Formulation of inference problem

To guarantee the identifiability of a network, we assume H is diagonal. For biological networks, this assumption means each measured node is perturbed by independent intrinsic or extrinsic noise. For this assumption to hold, at least B_e^1 is diagonal. In addition, without loss of generality, we assume qH is monic (meaning a direct perturbation of noise exists on measured nodes), which is satisfied by scaling the noise covariance matrix. As a result, we do not need other prior knowledge of transfer matrices, Q and P. The expression of a DSF can be rewritten as:

(5.8.1)

$$Y = F_u U + F_y Y + \hat{E}$$

where

$$F_u = (qH)^{-1} P$$
$$F_y = I - (qH)^{-1}(I - Q)$$
$$\hat{E} = q^{-1} E$$

An important fact is that since H is diagonal, transfer matrices F_y and F_u have the same zero structure as matrices Q and P, respectively. Therefore, F_y and F_u also reflect the network topology. Since the target network is sparse, F_y and F_u are sparse transfer matrices.

Remark 5.8.1: Note that transfer matrix H is strictly proper as indicated by (5.7.2). To deduce expression (5.8.1), we first rewrite DSF as $Y = QY + PU + (qH)(q^{-1}E)$. Since (qH) is proper and monic, F_y and F_u are strictly proper transfer matrices. In other words, (5.8.1) shows the present outputs depend on the past inputs and outputs. Hence, (5.8.1) can be used to deduce the predictor of the network.

To identify the parametrized model (5.8.1), one has to estimate the system order and model parameters of each transfer function in F_y and F_u. Since the order of these transfer functions is associated to the number of hidden nodes as well as the connectivity between hidden nodes and manifest nodes, model selection requires an exhaustive search of all possible combinations of system order. The estimation of model parameters leads to a highly nonlinear optimization problem. In addition, it is difficult to impose system stability during identification. To formulate a well-posed identification problem, we present (5.8.1) in a non-parametric way, where transfer functions are replaced by their impulse responses. For the ith node, we extract its subsystem from (5.8.1), which represents regulations from other nodes and inputs:

(5.8.2)

$$y_i(t) = \sum_{j=1}^{p}\sum_{k=1}^{\infty} h_{ij}^{y}(k)y_j(t-k) + \sum_{j=1}^{m}\sum_{k=1}^{\infty} h_{ij}^{u}(k)u_j(t-k) + e_i(t)$$

where h_{ij}^{y} and h_{ij}^{u} denote the impulse responses of transfer functions $\left[F_y\right]_{ij}$ and $[F_u]_{ij}$, respectively and are assumed to be stable (i.e. qH is minimum-phase). $e_i(t)$ is i.i.d. Gaussian noise with variance σ^2. Output y_i are the convolution of impulse responses with inputs and nodes.

To cast the system identification problem under the Bayesian framework, impulse responses are assumed to be independent zero mean Gaussian processes following the discussion in section 5.5. This scheme was originally applied in [113] to infer networks described by Granger causality. The covariance functions of the Gaussian processes are kernel functions that lead to a RKHS of causal and stable impulse responses. We use the discrete stable spline kernel as the covariance function. Consequently, the corresponding Bayesian probabilistic model is:

(5.8.3)

$$y_i(t) = \sum_{j=1}^{p}\sum_{k=1}^{\infty} X_{ij}^{y}(k)y_j(t-k) + \sum_{j=1}^{m}\sum_{k=1}^{\infty} X_{ij}^{u}(k)u_j(t-k) + e_i(t)$$

where $X_{ij}^{y}(k)$ and $X_{ij}^{u}(k)$ are sampled versions of zero mean Gaussian processes with the scaled stable spline kernel:

(5.8.4)

$$\bar{k}(t,s;\beta) = \left(\frac{e^{-\beta(t+s)}e^{-\beta\max(t,s)}}{2} - \frac{e^{-3\beta\max(t,s)}}{6}\right)$$

$$k(t,s;\lambda,\beta) = \lambda\bar{k}(t,s;\beta)$$

The kernel function is controlled by two hyperparameters, λ and β. β determines the decaying rate of impulse responses and λ takes charge of sparsity of the network. If λ_{ij}^{y} or λ_{ij}^{u} is 0, then $X_{ij}^{y}(k)$ or $X_{ij}^{u}(k)$ is discarded from the Bayesian model meaning the jth node or input does not control the ith node. We will see that estimating λ under the Bayesian framework is related to the Automatic Relevance Determination (ARD) technique, which promotes sparsity.

As discussed in section 5.5, the infinite sum in the Bayesian model (5.8.3) is defined in the sense of quadratic mean and results in a random variable at each time point t given past records of inputs and nodes. In addition, note that the auto-covariance of the Gaussian process is $k(t,t;\lambda,\beta) = \frac{\lambda e^{-3\beta t}}{3}$ which decays exponentially fast to 0 with time index t. Therefore, it is acceptable to use partial sum as an approximation to infinite sum in (5.8.3) for the sake of practical implementation. The Bayesian model (5.8.3) is approximated as:

(5.8.5)

$$y_i(t) = \sum_{j=1}^{p}\sum_{k=1}^{M} X_{ij}^{y}(k)y_j(t-k) + \sum_{j=1}^{m}\sum_{k=1}^{M} X_{ij}^{u}(k)u_j(t-k) + e_i(t)$$

where M is selected to be sufficiently large. From the view of model (5.8.2) in the functional setting, M is chosen so that the truncated impulse responses contain the most significant dynamical behavior of the system (i.e. $|h(t)|$ is small for $t > M$).

Assume time series data from indices 1 to N ($N \gg M$) for nodes and inputs are collected for inference. The objective is to estimate the Gaussian processes X^{y} and X^{u} at time points $\{t = 1,2,\cdots,M\}$.

5.9 Bayesian estimation of impulse responses

Under the Bayesian framework, impulse responses are estimated as the mean of the posterior distribution ($E(X(t)|y,u)$). To evaluate the posterior distribution, we define the following matrices and vectors associated to the ith node:

(5.9.1)

$$Y = \begin{bmatrix} y_N^i \\ \vdots \\ y_{M+1}^i \end{bmatrix},\ w = \begin{bmatrix} w_1 \\ \vdots \\ w_{p+m} \end{bmatrix}$$

$$A = \left[\begin{array}{ccc|ccc} y_{N-1:N-M}^{1} & \cdots & y_{N-1:N-M}^{p} & u_{N-1:N-M}^{1} & \cdots & u_{N-1:N-M}^{m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y_{M:1}^{1} & \cdots & y_{M:1}^{p} & u_{M:1}^{1} & \cdots & u_{M:1}^{m} \end{array}\right]$$

$$\sigma^2 = E\{e_i(t)^2\}$$

$$w_r = \begin{cases} \left[X_{ir}^{y}(1) \quad \cdots \quad X_{ir}^{y}(M)\right]' & \text{if } 1 \le r \le p \\ \left[X_{i(r-p)}^{u}(1) \quad \cdots \quad X_{i(r-p)}^{u}(M)\right]' & \text{if } p < r \le p+m \end{cases}$$

where $Y \in \mathbb{R}^{N-M}$, $w \in \mathbb{R}^{M(p+m)}$ and $A \in \mathbb{R}^{(N-M)\times M(p+m)}$.

Vector w is divided into p groups, each of which corresponds to the sampled impulse responses of a node or input. As the network topology is sparse, vector w is group sparse and its sparsity pattern reflects the network topology.

For the ith node, we have the likelihood function of (5.8.5) based on Bayes' rules:

(5.9.2)

$$\begin{aligned} & p(y_{M+1:N}^i|w,\sigma^2) \\ &= \prod_{j=0}^{N-M-1} p(y_{N-j}^i|y_{1:N-j-1}^i,w,\sigma^2) \\ &= \prod_{j=0}^{N-M-1} p(y_{N-j}^i|y_{N-j-T:N-j-1}^i,w,\sigma^2) \\ &= (2\pi\sigma^2)^{-\frac{N-M}{2}} exp\left(-\frac{1}{2\sigma^2}\|Y-Aw\|_2^2\right) \end{aligned}$$

where the dependence on the other nodes and inputs is suppressed for convenience.

The logarithm of the likelihood function is quadratic with respect to w. The prior of w is Gaussian because each group of w is an independent sampled Gaussian process:

(5.9.3)

$$p(w) = \prod_{j=1}^{p+m} p(w_j)$$

where

$$\begin{aligned} p(w_j) &\sim \mathcal{N}(w_j|0,K_j) \\ \left[K_j\right]_{ts} &= k(t,s;\lambda_j,\beta_j) \end{aligned}$$

As a result, the conditional posterior distribution of w is also Gaussian:

(5.9.4)

$$p(w|y,\sigma^2,\lambda,\beta) = |2\pi\Sigma|^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(w-\mu)'\Sigma^{-1}(w-\mu)\right\}$$

where

$$\begin{aligned} \Sigma^{-1} &= \frac{1}{\sigma^2}A'A + K^{-1} \\ \mu &= \frac{1}{\sigma^2}\Sigma A'Y \\ K &= blkdiag\{K_1,\dots,K_{p+m}\} \end{aligned}$$

Let $\beta = [\beta_1,\dots,\beta_{p+m}]'$ and $\lambda = [\lambda_1,\dots,\lambda_{p+m}]'$. Noise variance σ^2, and hyperparameters β and λ are unknown and remain to be determined.

5.10 Estimation of hyperparameters

Hyperparameters β and λ control the dynamics of impulse responses and sparsity of

network topology, respectively. A full Bayesian model introduces hyperpriors for these hyperparameters leading to posterior distribution $p(w,\sigma^2,\lambda,\beta,|y)$. To estimate impulse responses, hyperparameters are marginalized out from the model to achieve the marginal posterior distribution, $p(w|y)$. Nevertheless, $p(w|y)$ is intractable because the full Bayesian model is highly nonlinear with respect to these hyperparameters. Approximation techniques must be applied to tackle the intractable integral. There are two basic options: deterministic and stochastic approximations. We will consider stochastic approximations in the next chapter. In this chapter, we resort to the empirical Bayes method to approximate $p(w|y)$ with $p(w|y,\sigma^2,\lambda,\beta)$ analytically.

Following the discussion of section 4.3, to minimize the gap between the true and the approximate distributions, empirical Bayes optimizes hyperparameters by maximizing their likelihood function: $(\hat{\sigma}^2,\hat{\lambda},\hat{\beta}) = \arg\max_{\beta,\lambda,\sigma^2} p(y|\beta,\lambda,\sigma^2)$ (type II maximization) [71], [117]. By marginalizing out impulse responses, the likelihood function of hyperparameters is:

(5.10.1)

$$p(y|\sigma^2,\lambda,\beta) \propto \frac{1}{|\sigma^2 I + AKA'|} exp\left\{-\frac{1}{2}Y'(\sigma^2 I + AKA')^{-1}Y\right\}$$

Hence, the resulting optimization problem is as follows:

(5.10.2)

$$\left(\hat{\sigma}^2,\hat{\lambda},\hat{\beta}\right) = \arg\min_{\beta,\lambda,\sigma^2} Y'(\sigma^2 I + AKA')^{-1}Y + \log|\sigma^2 I + AKA'|$$

subject to

$$\beta,\lambda,\sigma^2 \geq 0$$

Note that the cost function in (5.10.2) is highly nonlinear and non-convex. The EM algorithm is commonly used to solve (5.10.2) where impulse responses w are treated as the latent random variable. To further simplify notation, let $\theta = (\beta,\lambda,\sigma^2)$ and $Z = w$. In the E step, function $Q(\theta|\theta^{t-1}) = E_{Z|Y,\theta^{t-1}}[\log p(Y,w|\beta,\lambda,\sigma^2)]$ of the current iteration is calculated using θ^{t-1} from the last iteration. Note that:

(5.10.3)

$$\begin{aligned} Q(\theta|\theta^{t-1}) &= L_1(\sigma^2) + L_2(\beta,\lambda) \\ L_1(\sigma^2) &= \int \log p(Y|w,\sigma^2)\, p(w|\beta^{t-1},\lambda^{t-1})dw \\ L_2(\beta,\lambda) &= \int \log p(w|\beta,\lambda)\, p(w|\beta^{t-1},\lambda^{t-1})dw \end{aligned}$$

where μ_i^{t-1} is the ith block of μ^{t-1} corresponding to a node or input. Σ_i^{t-1} is the ith diagonal block of Σ^{t-1}. $\left[\overline{K}_j\right]_{ts} = \overline{k}(t,s;\beta_j)$ is the stable spline kernel without scaling.

$$\begin{aligned} L_1(\sigma^2) &= -\frac{N-M}{2}\log\sigma^2 + \int \left(-\frac{1}{2\sigma^2}\|Y - Aw\|_2^2\right) p(w|\beta^{t-1},\lambda^{t-1})dw \\ &= -\frac{N-M}{2}\log\sigma^2 + E\left\{-\frac{1}{2\sigma^2}(Y'Y - 2Y'Aw + w'A'Aw)\right\} \end{aligned}$$

$$= -\frac{N-M}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\{Y'Y - 2Y'AE(w) + E[tr(A'Aww')]\}$$
$$= -\frac{N-M}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\{Y'Y - 2Y'A\mu^{t-1} + tr[A'A(\Sigma^{t-1} + \mu^{t-1}\mu^{t-1\prime})]\}$$
$$= -\frac{N-M}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\{Y'Y - 2Y'A\mu^{t-1} + \mu^{t-1\prime}A'A\mu^{t-1} + tr[A'A\Sigma^{t-1}]\}$$
$$= -\frac{N-M}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\{\|Y - A\mu^{t-1}\|_2^2 + tr[A'A\Sigma^{t-1}]\}$$

$$L_2(\beta,\lambda) = \int \left(\log|2\pi K|^{-\frac{1}{2}} - \frac{1}{2}w'K^{-1}w\right)p(w|\beta^{t-1},\lambda^{t-1})dw$$
$$= -\frac{1}{2}\log|K| - \frac{1}{2}E\{tr[K^{-1}ww']\}$$
$$= -\frac{1}{2}\log|K| - \frac{1}{2}tr[C^{-1}(\Sigma^{t-1} + \mu^{t-1}\mu^{t-1\prime})]$$
$$= -\frac{M}{2}\sum_{i=1}^{p+m}\log\lambda_i - \frac{1}{2}\sum_{i=1}^{p+m}\log|\bar{K}_i| - \frac{1}{2}\sum_{i=1}^{p+m}\lambda_i^{-1}tr[\bar{K}_i^{-1}(\Sigma_i^{t-1} + \mu_i^{t-1}\mu_i^{t-1\prime})]$$

In the M step, $\theta^t = \arg\max_\theta Q(\theta|\theta^{t-1})$ is solved. According to (5.10.3), hyperparameters can be estimated independently. The optimal σ^2 has a closed-form solution as $(\sigma^2)^t = \frac{1}{(N-M)}\{\|Y - A\mu^{t-1}\|_2^2 + tr[A'A\Sigma^{t-1}]\}$. Optimizing β and λ can be decomposed into small sub-problems.

(5.10.4)

$$(\lambda_i^t, \beta_i^t) = \arg\min_{\lambda_i,\beta_i} M\log\lambda_i + \log|\bar{K}_i| + \lambda_i^{-1}tr[\bar{K}_i^{-1}(\Sigma_i^{t-1} + \mu_i^{t-1}\mu_i^{t-1\prime})]$$

The optimal solution for λ_i^t is determined by β_i^t as:

(5.10.5)

$$\lambda_i^t = \frac{1}{M}tr[\bar{K}_i^{-1}(\Sigma_i^{t-1} + \mu_i^{t-1}\mu_i^{t-1\prime})]$$

Insert (5.10.5) back into (5.10.4), we have

(5.10.6)

$$\beta_i^t = \arg\min_{\beta_i} M\log tr[\bar{K}_i^{-1}(\Sigma_i^{t-1} + \mu_i^{t-1}\mu_i^{t-1\prime})] + \log|\bar{K}_i|$$

The EM algorithm is summarized as follows.

Algorithm 1 The kernel method equipped with EM
1: ***Initialization***: set $((\sigma^2)^0, \beta^0, \lambda^0)$
2: ***For*** $t = 1: MAX$ do
3: Update σ^2 as

$$(\sigma^2)^t = \frac{1}{(N-M)}\{\|Y - A\mu^{t-1}\|_2^2 + tr[A'A\Sigma^{t-1}]\}.$$

4: Update β independently by solving

$$\beta_i^t = \arg\min_{\beta_i} M\log tr[\overline{K}_i^{-1}(\Sigma_i^{t-1} + \mu_i^{t-1}\mu_i^{t-1\prime})] + \log|\overline{K}_i|$$

5: Calculate $\overline{K}_i$ using β_i^t and update λ independently as

$$\lambda_i^t = \frac{1}{M} tr[\overline{K}_i^{-1}(\Sigma_i^{t-1} + \mu_i^{t-1}\mu_i^{t-1\prime})]$$

6: ***End for***

One advantage of EM is that problem (5.10.2) is decomposed into a series of small-scale problems. Hyperparameters are updated independently in each iteration. Hence, the algorithm scales well with large networks. However, the algorithm may suffer from numerical instability in practice. In each iteration, the inversion of $\overline{K}_i$ is calculated, which can be ill-conditioned for large values of β_i. It was shown that some kernel functions (e.g. DC and TC kernels) have closed-form inversion [201]. Unfortunately, it is not the case for the second order stable spline kernel [202].

Problem (5.10.2) can also be solved directly. Since calculating the derivative of the cost function with respect to β is non-trivial, derivative-free algorithms such as the simplex search method are preferred. The resulting algorithm is more numerically stable since the calculation of $\overline{K}_i^{-1}$ is avoided. In addition, the algorithm converges faster than the EM algorithm. Nevertheless, the algorithm no longer scales with the size of the target network. Also, it easily gets trapped at local optimal solutions. The estimation of hyperparameter β is relatively robust to local optima [74]. However, the evaluation of λ is very sensitive to the suboptimal solutions because its sparsity pattern determines the network topology.

Another aspect that must be noticed is the estimation of noise variance σ^2. In deterministic approximations, even the true value of the noise variance may not be the best choice for approximating the posterior distribution. That means σ^2 is no longer the noise variance but rather a tuning variable to some extent.

Compared with the type II maximization of SBL in the last chapter, the cost function differs in the choice of the kernel function. For SBL, the kernel function is:

(5.10.7)

$$k(t,s) = \begin{cases} \beta_t & \text{if } t = s \\ 0 & \text{if } t \neq s \end{cases}$$

where $\beta_r \geq 0$

Impulse responses in the RKHS constructed by kernel (5.10.7) are not stable according to Theorem 5.6.2. SBL allows the highest degree of freedom to estimate unknown model parameters by ignoring correlations among them. As the predictor of ARX models is a FIR system (that is inherently stable), impulse responses do not necessarily decay exponentially. Hence, correlations among elements of FIRs are weak and SBL is suitable to estimate ARX models. For IIR systems (e.g. the predictor of DSFs), the property of system stability causes strong correlations among impulse responses at different time points. In particular, impulse responses decay exponentially fast to zero for stable systems. As a result, to enforce system stability, the kernel function used to construct the RKHS for stable impulse responses must be

bounded by exponential functions.

5.10.1 Selection of the inferred links

Under the framework of empirical Bayes, hyperparameter λ introduces the effect of the ARD technique. If λ_i is zero, the corresponding node or input is removed from the model. In practice, one does not expect estimated λ_i to be exactly zero due to numerical errors. A conventional way is to use a threshold to rule out small λ_i. However, it is difficult to determine the most proper threshold. In addition, since the stable spline kernel is used, the effect of the ARD technique also depends on hyperparameter β. Hence, instead of evaluating the contribution of a node or input to system dynamics based on λ only, we refer to the estimated impulse responses directly. A node or input is less important if its corresponding impulse responses are small. In this section, we propose a heuristic way to select the inferred nodes.

To begin with, we rank the importance of nodes and inputs in an ascent sequence, according to the norm of their estimated impulse responses. Nodes and inputs are then removed one-by-one from the bottom of the sequence. In each iteration, the corresponding cost function in (5.10.2) is evaluated. If the value of the cost function is decreased, the removal is accepted. The iteration stops when all nodes and inputs are removed from the model. The algorithm to perform the above procedure is as follows.

Algorithm 2 Selection of the inferred links

1: Solve problem (5.10.2) to get estimated hyperparameters $\hat{\sigma}^2, \hat{\beta}, \hat{\lambda}$.

2: Calculate $f(\hat{\sigma}^2, \hat{\beta}, \hat{\lambda})$ where $f(\sigma^2, \beta, \lambda) = Y'(\sigma^2 I + AKA')^{-1}Y + \log|\sigma^2 I + AKA'|$.

2: Estimate impulse responses as $\widehat{w} = \frac{1}{\sigma^2}\Sigma A'Y$ and calculate the norm for each group.

3: Rank impulse responses and set the threshold as $R = sort\{[\|\widehat{w}_1\|_2 \quad \cdots \quad \|\widehat{w}_{p+m}\|_2]\}$ where $sort$ is the function in Matlab.

2: ***For*** $t = 1{:}\,p + m - 1$ do

3: Define the set of nodes and inputs to be removed as

$$I = \{i | \|\widehat{w}_i\|_2 \le R_t\}$$

4: Set $\hat{\lambda}_i^t$ to be zero if it is included in set I

$$\hat{\lambda}_i^t = \begin{cases} 0 & \text{if } i \in I \\ \hat{\lambda}_i & \text{if } i \notin I \end{cases}$$

5: Accept $\hat{\lambda}_i^t$ if the cost function is decreased

$$\hat{\lambda} = \hat{\lambda}^t \text{ if } f(\hat{\sigma}^2, \hat{\beta}, \hat{\lambda}^t) \le f(\hat{\sigma}^2, \hat{\beta}, \hat{\lambda})$$

6: ***End for***

7: Calculate the final estimation of impulse responses using $\hat{\sigma}^2, \hat{\beta}, \hat{\lambda}$ as $\widehat{w} = \frac{1}{\sigma^2}\Sigma A'Y$

5.11 Models with measurement noise

In previous sections, we assume there is no or small measurement noise. If measurement

noise is not negligible, it should be considered in the proposed model. We consider a state space model in the innovation form:

(5.11.1)

$$x(t+1) = Ax(t) + B_u u(t) + B_e e(t)$$
$$y(t) = Cx(t) + e(t)$$

where $e(t)$ contains information of both process and measurement noise. As the prediction error of the model, $e(t)$ is assumed to be white Gaussian noise.

Following the same protocol, the dynamical structure function of (5.11.1) is:

(5.11.2)

$$Y = QY + PU + (I - Q + H)E$$

where Q, P and H are strictly proper transfer matrices from (5.7.3).

Model (5.11.2) can be rewritten as:

(5.11.3)

$$Y = F_u U + F_y Y + E$$

where

$$F_u = (I - Q + H)^{-1} P$$
$$F_y = I - (I - Q + H)^{-1}(I - Q)$$

To see (5.11.3) is physically realizable, we need to verify that $(I - Q + H)^{-1}$ is causal. The state space realization of $F = (I - Q + H)$ is:

(5.11.4)

$$F \sim \begin{bmatrix} A_F & B_F \\ C_F & D_F \end{bmatrix}$$

$$A_F = \begin{bmatrix} A_Q & 0 & 0 \\ 0 & A_H & 0 \\ 0 & 0 & A_I \end{bmatrix},\ B_F = \begin{bmatrix} B_Q \\ B_H \\ 0 \end{bmatrix},\ C_F = [C_Q \quad C_H \quad 0],\ D_F = I$$

where state space realizations for Q, H and I are:

$$Q \sim \begin{bmatrix} A_Q & B_Q \\ C_Q & 0 \end{bmatrix},\ H \sim \begin{bmatrix} A_H & B_H \\ C_H & 0 \end{bmatrix},\ I \sim \begin{bmatrix} A_I & 0 \\ 0 & I \end{bmatrix}$$

Since D_F is non-singular, F^{-1} exists and is a causal system [171] with the state space realization:

$$F^{-1} \sim \begin{bmatrix} A_F - B_F C_F & -B_F \\ C_F & D_F \end{bmatrix}$$

Moreover, F_u and F_y are both strictly proper because $F_u \to 0$, $F_y \to 0$ as $q \to \infty$. Consequently, the predictor of (5.11.2) can be easily derived from (5.11.3).

The framework discussed in this chapter can also be used to identify model (5.11.3). Nevertheless, transfer matrices F_u and F_y are no longer sparse but full. Hence, the ARD technique does not apply here. In this case, how to promote network sparsity requires further research.

5.12 Conclusion

This chapter applies state space models to describe sparse networks. Considering only a small number of biological units can be measured in practice, DSF models are used to

represent networks consisting of observable nodes while hidden nodes are encoded through transfer functions. DSFs are expressed in a non-parametric way using impulse responses so that inference can be carried out without prior knowledge of the number of hidden nodes and their biological interpretations. Following the framework of kernel methods, the stable spline kernel is used to establish a functional space (RKHS) of causal and stable impulse responses. The identification problem is recast in the Bayesian framework. Impulse responses are assumed to be independent Gaussian processes whose covariance function is the stable spline kernel. The empirical Bayes technique is employed to approximate the intractable marginal posterior distribution of impulse responses. Type II maximization is used to optimize hyperparameters of the kernel function that determine network topology and internal dynamics.

One advantage of the kernel method is that the challenging problem of model selection is avoided. By using the non-parametric identification framework, the combinatorial search for system order of transfer functions is avoided. More importantly, the method guarantees to generate a stable system. ARD is naturally embedded into empirical Bayes to promote sparse network topologies.

The main drawback of the kernel method is that the resulting optimization problem is highly nonlinear. Hence, normally only local optima can be achieved. As the sparsity pattern of the solution determines network topology, local optima seriously influence the performance of the algorithm. In addition, estimating the noise variance is problematic because it behaves as a tuning variable rather than a variable controlling statistical properties of process noise.

Further developments are three-fold. One is to find a more advanced algorithm that can encourage a global search of the optimal solution. The second is to develop a better strategy to evaluate the noise variance. The difficulty of its estimation is an inherent problem of deterministic approximation techniques. Hence, stochastic approximations can be a potential solution which explores the true Bayesian model directly without analytical approximations. Finally, it is important to find a way to promote network sparsity with the existence of measurement noise.

Chapter 6.

Sparse network inference using reversible jump Markov chain Monte Carlo

Stochastic dynamical models have been used to describe topology and internal dynamics of networks. Model parameters are assumed to be random quantities to reflect model uncertainty. By incorporating prior knowledge, full Bayesian models are established to present posterior belief of dynamical models given time series data. These probabilistic models can be used for system identification, model comparison and prediction of future network behaviors. However, manipulation of Bayesian models (such as marginalization and normalization) is typically intractable, due to highly complex model expressions. Hence, approximation techniques must be used to tackle the problem. Previous chapters are focused on empirical Bayes as a deterministic approximation method to analytically approximate a distribution. This technique has been applied in GESBL and the kernel method where a Gaussian distribution is used to approximate intractable true distributions. The usage of Gaussian distributions greatly simplifies Bayesian estimation that requires to perform marginalization and calculate expectation. It has been shown that empirical Bayes effectively imposes sparse network topologies. However, since the true Bayesian model is replaced by an approximate one, it is unlikely to have an accurate evaluation of model uncertainty. Ideally, we would like to evaluate the probability of all possible network topologies. In addition, deterministic approximations require us to solve an optimization problem that can be highly nonlinear. Local optimal solutions can seriously degrade the ability of algorithms to detect network topology. Finally, estimation of noise variance is problematic since noise variance is treated as a tuning variable to optimize the approximate distribution. An accurate estimation of noise variance helps algorithms filter out process noise, thus producing reliable inference results. Hence, without strong ability to explore statistical properties of process noise, the performance of the developed methods degrades as process noise increases.

This chapter treats network topology as a new random quantity and embeds it into the full Bayesian model. The full Bayesian model is established based on DSFs. Stochastic approximations are applied to explore the probabilistic model directly. A sampling method called reversible jump Markov chain Monte Carlo (RJMCMC) is used to encourage a global

search of network topology by traversing among parameter subspaces of different dimensionality, each of which corresponds to a particular network topology. Metropolis-Hastings-within-Partially Collapsed Gibbs sampler (MH-within-PCG) is carefully designed to efficiently draw samples from the full Bayesian model. By exploring the statistical property of process noise via sampling, the developed method is robust to noise perturbations. In addition, the confidence of the inferred network topology is clearly shown by the empirical marginal posterior distribution. Simulations indicate that RJMCMC outperforms SBL, GSBL and the kernel method on identifying multivariable ARX models. It is also superior to the kernel method when dealing with DSF models. Nevertheless, how to reduce its computational cost remains to be an open question.

Next is an outline of this chapter. Sections 6.1 discusses basic Markov chain Monte Carlo (MCMC) methods. Sections 6.2 and 6.3 illustrate details of MH-within-PCG samplers and RJMCMC techniques. Section 6.4 briefly reviews DSFs to describe sparse networks and handle hidden nodes. Section 6.5 formulates the network reconstruction problem. Section 6.6 constructs the full Bayesian model of DSFs and section 6.7 deduces a MH-within-PCG sampler with the fixed network topology. Section 6.8 applies RJMCMC to draw samples from the full Bayesian model. Section 6.9 discusses the computational cost of the sampler. Finally, section 6.10 compares different approaches via Monte Carlo simulations.

6.1 MCMC approach

For most real-world applications, exact inference is typically intractable and requires approximations [90]. Previous chapters apply deterministic methods to approximate ground truth distributions analytically using empirical Bayes. Another way to explore probabilistic models are Monte Carlo techniques. They are approximate inference methods based on numerical sampling [90], [40]. Stochastic approximation techniques such as Markov chain Monte Carlo (MCMC) are widely used in practice where drawn samples are guaranteed to asymptotically converge to the target distribution. These samples can be used to evaluate intractable marginal probabilistic models. They can also be used to estimate expectation of random variables that cannot be calculated in a closed form.

MCMC is a very powerful framework, which allows samples to be drawn from a large class of distributions. In particular, MCMC can handle complex distributions that cannot be sampled directly. For example, MCMC can sample a distribution that is known up to a constant. Moreover, MCMC scales well with the dimensionality of the sample space [90]. The basic idea of MCMC schemes is to construct a Markov chain so that the distribution of the Markov state converges to the target distribution as the length of the chain grows.

6.1.1 Markov chain

Before discussing more details, for the sake of completeness, we briefly review some basic concepts of Markov chain. A thorough treatment can be found in books and rich literatures, for example, [18], [40], [90], [162], [165], to name a few.

A first-order discrete time Markov chain is a stochastic process $\{X_t, t \in \{1,2,3\ldots\}\}$ (Markov state) taking values in an arbitrary state space ($\mathbb{R}^n$ in this chapter) so that $p(x_{t+1}|x_1, x_2, \ldots, x_t) = p(x_{t+1}|x_t)$ where x denotes the value of the state and $p(\cdot)$

represents the probability density function. That is, the future of the Markov state is independent on the past given the current state. Hence, a Markov chain is fully specified provided the probability distribution of the initial state and the conditional probabilities for subsequent states called transition probabilities (kernels) [165]. (Note that kernels of a Markov chain and kernel functions in the last chapter are different concepts.) In this chapter, we only consider homogeneous Markov chains whose transition kernels are independent on time index t.

For a general state space S, a transition kernel is expressed as $P(x, B) = \Pr(X_t \in B | X_{t-1} = x)$ indicating the probability of moving from state x into set B, where $x \in S$ and B is a measurable set in S. For each fixed x, map $B \to P(x, B)$ is a probability measure. For each fixed B, map $x \to P(x, B)$ is a measurable function. The product of two transition kernels is defined as $P_1 P_2(x, A) = \int P_1(x, dy) P_2(y, A)$ [165], which can be interpreted as the probability of moving from state x into set A by executing transition 1 (with kernel P_1) and 2 (with kernel P_2) sequentially.

The marginal distribution of the state at time $t+1$ can be expressed as $\pi_{t+1}(A) = \int P(x, A) \pi_t(dx)$ where $\pi_t(\cdot)$ is the distribution (probability measure) of X_t. A distribution $\pi^\star(\cdot)$ is said to be invariant or stationary with respect to a Markov chain if $\pi^\star(B) = \int P(x, B) \pi^\star(dx)$ for any measurable set B. That means if a Markov chain starts from the initial state with the invariant distribution, all the subsequent states will have the same marginal distribution as the initial one. A sufficient condition for ensuring the target distribution $\pi^\star(\cdot)$ is invariant is to select a transition kernel so that 'detailed balance' is satisfied: $\int_A P(x, B) \pi^\star(dx) = \int_B P(x, A) \pi^\star(dx)$ for any measurable sets A and B or $p^\star(x') p(x|x') = p^\star(x) p(x'|x)$ where $p^\star(\cdot)$ and $p(\cdot|x)$ denote density functions of Lebesgue measure with respect to probability measures $\pi^\star(\cdot)$ and $P(x, \cdot)$, respectively [165], [203]. That is, the probability of moving to set B or state x from set A or state x' is equal to from set A or state x' to set B or state x. In other words, the Markov chain is reversible.

In practice, a single update mechanism (characterized by a transition kernel) may not be sufficient to draw samples from the entire state space. If update mechanisms are not well designed, the produced samples cannot accurately represent statistical properties of the target random variable, thus leading to a biased estimation of its distribution. Multiple update mechanisms can be combined to yield a more efficient sampler. For example, in each iteration, Gibbs sampling contains several update steps, each of which only updates one variable with others fixed. There are two major ways to combine update mechanisms, composition and mixing [165]. The invariant distribution shared by multiple update mechanisms is preserved after combination if certain rules are obeyed. These rules are very important for deducing and validating novel samplers in following sections.

The composition method integrates multiple update mechanisms into a chain so that these update mechanisms are executed sequentially. For m update mechanisms U_1, U_2, …, U_m with transition kernels P_1, P_2, …, P_m that all preserve the same invariant distribution, they are implemented in the sequence $U_1 \longrightarrow U_2 \longrightarrow \cdots \longrightarrow U_m$. The resulting transition kernel of the Markov chain turns out to be the product of individual kernels $P(x, A) = P_1 P_2 \cdots P_m(x, A)$ [165]. The invariant distribution is preserved in this case.

The mixing method executes one of the update mechanisms at a time randomly. Suppose

there are m update mechanisms U_1, U_2, …, U_m accompanied by fixed probabilities p_1, p_2, …, p_m where $\sum_{i=1}^{m} p_i = 1$. At each iteration, an update mechanism U_i is selected with probability p_i to generate the next Markov state. The resulting transition kernel of the Markov chain in this case is a convex combination of these transition kernels $P(x, A) = \sum_{i=1}^{m} p_i P_i(x, A)$ [165]. In this way, the invariant distribution is preserved.

To draw samples from the target distribution, we require the target distribution to be invariant for the constructed Markov chain, which means the transition kernel must be formulated to meet the 'detailed balance' condition. In addition, as $t \longrightarrow +\infty$, the marginal distribution of the Markov state, $p(x_t)$ must converge to the target distribution, irrespective of the choice of the initial distribution, $p(x_1)$. This property is called 'ergodicity'. For a chain to be ergodic, it must be irreducible (meaning it is possible to visit all states from any current state) and aperiodic (no cycles) [180], [165]. A homogeneous Markov chain is ergodic under some mild conditions on the invariant distribution and transition kernel [180].

6.1.2 Metropolis-Hastings algorithm (MH) and Gibbs sampler

MH algorithms and Gibbs sampling are two widely used MCMC methods to draw samples [204], [165], [205]. Since Gibbs sampling can be treated as a special case of MH, we introduce MH first.

MH algorithms can draw samples from a distribution that is known up to a normalization constant. Assume we want to draw samples from distribution $p(x) = \frac{1}{Z}\tilde{p}(x)$ where normalization constant z is unknown. In MH algorithms, with current state x_t, a sample x^{prop} is drawn from proposal distribution $q(x^{prop}|x_t)$ and then accepted with probability $A(x^{prop}|x_t) = \min\left\{1, \frac{\tilde{p}(x^{prop})q(x_t|x^{prop})}{\tilde{p}(x_t)q(x^{prop}|x_t)}\right\}$ [206]. If the proposal is accepted, $X_{t+1} = x^{prop}$. If not, $X_{t+1} = x_t$. It can be seen that the transition probability of the Markov chain using MH algorithms is $p(x|x') = q(x|x')A(x|x')$ and, hence, 'detailed balance' is satisfied [203]. The choice of proposal distributions is crucial, which determines the statistical property of Markov chains. An inadequate choice can result in poor performance of designed Monte Carlo samplers (e.g. low convergence speed, high rejection rate) [206].

For Gibbs sampling, consider the target density function, $p(x^1, x^2, \dots, x^m)$ from which we want to draw samples. The Markov state is $x = (x^1, x^2, \dots, x^m)$. At each step, Gibbs sampling draws a sample of one variable from its full conditional distribution conditioned on the latest update of other variables [207], [72]. At step t, if $m = 3$, x_t^1 is updated by x_{t+1}^1 which is sampled from distribution $p(x^1|x_t^2, x_t^3)$. Next, x_t^2 is replaced by x_{t+1}^2 sampled from distribution $p(x^2|x_{t+1}^1, x_t^3)$. Finally, x_t^3 is replaced by x_{t+1}^3 sampled from distribution $p(x^3|x_{t+1}^1, x_{t+1}^2)$. The above procedure is then executed cyclically. It is important to realize that the Markov chain is formulated as (x_1^1, x_1^2, x_1^3), (x_2^1, x_2^2, x_2^3), …, (x_N^1, x_N^2, x_N^3) instead of (x_1^1, x_1^2, x_1^3), (x_2^1, x_1^2, x_1^3), (x_2^1, x_2^2, x_1^3), (x_2^1, x_2^2, x_2^3), etc. Each intermediate step of Gibbs sampling can be regarded as an MH sampling with acceptance probability always being 1 [165]. The target distribution is the invariant of each intermediate sampling step and, therefore, of the whole Markov chain according to the property of composition of transition kernels. Constructing a Gibbs sampler is straightforward. However, the applicability of Gibbs sampling highly depends on whether conditional distributions can be sampled directly or not

[90].

6.2 MH-within-PCG sampler

In practice, Gibbs and MH sampling are barely used directly. Rather, their variants are applied accordingly to handle complex distributions. For example, blocked Gibbs samplers are a minor variant of traditional Gibbs samplers which groups two or more variables and samples from their joint conditional distribution, by which a better convergence property is achieved [208]. Another example is the single-site updating MH sampling where only one component of the current Markov state is updated at a time so that the design of proposal distributions is simplified [209], [203].

Gibbs and MH sampling can also be combined to build up a more powerful and efficient sampler. If the conditional distributions of some sampling steps of a Gibbs sampler cannot be sampled directly, we can replace these steps with MH sampling schemes thus generating a hybrid sampler (MH-within-Gibbs sampler) [210]. It is also known that by marginalizing out some random variables of sampling steps, the convergence property of a sampler can be improved [210], [211]. Modified Gibbs samplers based on this idea are called partially collapsed Gibbs sampler (PCG) [212], [211]. It is called Metropolis-Hastings within partially collapsed Gibbs sampler (MH-within-PCG) for modified hybrid samplers [210]. In this chapter, we will use marginalization to improve the performance of samplers.

Introducing marginal distributions into a Gibbs sampler to form a PCG is nontrivial. The distribution of a sampling step cannot be marginalized directly. Otherwise, the invariant distribution of the Markov chain can be changed. However, this issue was not sufficiently aware of in much of the previous research. It has been shown that some rules for marginalization must be followed to preserve the invariant distribution [210]. To reduce the number of conditioned random variables, steps called marginalization, permutation and trimming are executed in sequence [211]. Marginalization means to move components of Markov states from being conditioned on to being sampled. For example, we can replace sampling x from $p(x|y,z)$ with sampling (x,y) from $p(x,y|z)$, which does not change invariant distribution $p(x,y,z)$ of a Gibbs sampler. Permutation means to switch the order of sampling steps. Finally, trimming is used to discard a subset of components to be sampled but not conditioned on in the next step. For instance, if variable y sampled from $p(x,y|z)$ is not conditioned on in the next step (e.g. sampling $p(y|x,z)$), $p(x|z)$ can be sampled instead. By trimming, the transition kernel is not changed so that the invariant distribution is preserved.

It is important to realize that sampling steps of a PCG sampler cannot be replaced by their MH counterparts directly because it may change the invariant distribution. A MH-within-PCG sampler must be derived from the original MH-with-Gibbs sampler following similar rules of marginalization, permutation and trimming [210]. The key point is that a full MH step of a MH-with-Gibbs sampler can be replaced by a reduced MH step followed immediately by a direct draw from the complete conditional distribution of those reduced quantities [210]. For instance, the MH step to sample $p(x|y,z)$ in a MH-with-Gibbs sampler can be replaced by the reduced MH step to sample $p(x|y)$ followed immediately by sampling $p(z|x,y)$. After this replacement, sampling steps can be reordered and redundant quantities can be trimmed

out following the rule of trimming.

In this chapter, we apply a MH-within-PCG sampler to draw samples from the full Bayesian model of the target network. Gibbs samplers are adopted as basic samplers for sampling. MH sampling is inevitably used since not all the conditional distributions can be sampled directly. In addition, we intend to marginalize out components to be sampled as much as possible in order to improve convergence speed, which leads to MH-within-PCG samplers.

6.3 Reversible jump Markov chain Monte Carlo

Traditional MCMC is used to draw samples from a joint distribution of random variables whose dimension (dimensionality of the sample space) is fixed. There are cases where the dimension of random variables varies. Under this circumstance, MCMC samplers must jump between parameter subspaces of different dimensionality in order to explore the target distribution sufficiently. Reversible jump Markov chain Monte Carlo (RJMCMC) was designed for this purpose [126], [203], [205], [213]. RJMCMC also generalizes normal MH algorithms.

RJMCMC was originally developed for Bayesian model selection [126], where the dimension of model parameters varies in accordance to system order. Later, RJMCMC was extended to other applications including optimization [214], [215], machine learning [216], [174] and system identification [180], [217]–[219].

Assume we have a countable collection of Bayesian models $\{\mathcal{M}_k, k \in \{1,2,3\ldots\}\}$. Each model $\mathcal{M}_k$ is characterized by a parameter vector, $\theta^k \in \mathbb{R}^{d_k}$, where the dimension, d_k may differ from model to model. The random variable to be sampled is expressed as $x = (k, \theta^k)$ which lies in the space, $S_k = \{k\} \times \mathbb{R}^{d_k}$ given k. Hence, the entire parameter space of x is $S = \bigcup_{k\in\{1,2,3,\ldots\}} S_k$ [203].

Suppose $\pi(x)$ is the probability density function of interest. To draw samples from $\pi(x)$, a reversible Markov chain $\{X_t, t \in \{1,2,3\ldots\}\}$ is constructed with the invariant distribution to be $\pi(x)$. Each Markov state X_t is composed of two components, k_t and θ_t^k where k_t is the model index and θ_t^k is the corresponding unknown model parameter. To traverse across the combined parameter space S, different types of moves are designed, among which only one move is selected randomly for each transition. Transition kernels are carefully designed so that 'detailed balance' is achieved for each move type. The resulting transition kernel of the Markov chain is the mixing of the transition kernels of all moves. Consequently, invariant distribution $\pi(x)$ is preserved [126].

Let (k, θ) be the current state, X_t of the Markov chain where $\theta \in \mathbb{R}^{d_k}$. Based on the designed moves, the probability to traverse from current model k to the next one k' is $p_{kk'}$, where $\sum_{k'} p_{kk'} = 1$. Note that if $k' = k$, only model parameters are updated in the next state. In addition, it is possible that not all the models can be reached in the next state from the current state. Reachability of a new state is determined by the designed moves. Given the proposed k' with probability $p_{kk'}$, $\theta' \in \mathbb{R}^{d_{k'}}$ is generated as the proposal of the next state. One way to generate θ' is to first produce a random quantity, U with probability density $pr_{kk'}(u|\theta)$ and then apply a deterministic map of previous state θ along with U. As a result, $\theta' = g_{1kk'}(\theta, U)$ where $U \in \mathbb{R}^{d_{kk'}}$ and $g_{1kk'}: \mathbb{R}^{d_k + d_{kk'}} \longrightarrow \mathbb{R}^{d_{k'}}$ is a deterministic map. The proposal, $X^{prop} = (k', \theta')$ is then accepted with probability $A_{kk'}(\theta'|\theta)$. If accepted, $X_{t+1} = X^{prop}$. If not, $X_{t+1} = X_t$ [203].

It is required that for the move from (k,θ) to (k',θ') and the reverse move from (k',θ') to (k,θ), their corresponding proposals (θ,U) and (θ',U') must have equal dimension. This is called 'dimension matching': $d_k + d_{kk'} = d_{k'} + d_{k'k}$ [126]. In addition, there must exist a deterministic map, $g_{2kk'}: \mathbb{R}^{d_k+d_{kk'}} \longrightarrow \mathbb{R}^{d_{kk'}}$ such that $(\theta',U') = g_{kk'}(\theta,U) = (g_{1kk'}(\theta,U), g_{2kk'}(\theta,U))$ where map $g_{kk'}$ is bijective and differentiable [203].

Finally, to achieve 'detailed balance', the acceptance probability is designed according to:

(6.3.1)

$$\pi(k,\theta)p_{kk'}pr_{kk'}(U|\theta)A_{kk'} = \pi(k',\theta')p_{k'k}pr_{k'k}(U'|\theta')A_{k'k}(\theta|\theta)\left|\frac{\partial g_{kk'}(\theta,U)}{\partial\theta\partial U}\right|$$

where

$$\theta' = g_{1kk'}(\theta,U)$$
$$U' = g_{2kk'}(\theta,U)$$

As a result, the acceptance probability equates to:

(6.3.2)

$$A_{kk'}(\theta'|\theta) = \min\left\{1,\ \frac{\pi(k',\theta')p_{k'k}pr_{k'k}(U'|\theta')}{\pi(k,\theta)p_{kk'}pr_{kk'}(U|\theta)}\left|\frac{\partial g_{kk'}(\theta,U)}{\partial\theta\partial U}\right|\right\}$$

6.4 Model formulation

We use state space model (5.7.1) to describe a sparse network. Considering some nodes of the network are not observable, DSF models are adopted to present the network of measurable nodes whereas hidden nodes are encoded by transfer functions. The model is expressed in the same way as chapter 5:

(6.4.1)

$$Y = QY + PU + HE$$

$Y \in \mathbb{R}^p$ represent observed nodes of the network. $U \in \mathbb{R}^m$ denote inputs. $E \in \mathbb{R}^q$ are i.i.d. Gaussian noise with zero mean and a diagonal covariance matrix. Q, P and H are strictly proper transfer matrices which represent internal dynamics of the network. The topology of the network is indicated by the zero structure of matrices Q and P where Q reflects causal interactions among nodes and P shows how inputs enter the network.

To guarantee the one-to-one correspondence between a DSF and an input-output map, H is assumed to be diagonal according to Theorem 5.7.1. Moreover, without loss of generality, qH is assumed to be monic and minimum-phase in order to derive the predictor of (6.4.1).

6.5 Problem formulation

The DSF in (6.4.1) can be rewritten as:

(6.5.1)

$$Y = F_u U + F_y Y + \hat{E}$$

where q denotes the time shift operator that $y(t+1) = qy(t)$ and:

$$F_u = (qH)^{-1}P$$

$$F_y = I - (qH)^{-1}(I - Q)$$
$$\hat{E} = q^{-1}E$$

Since H is diagonal, transfer matrices F_y and F_u have the same zero structure as matrices Q and P, respectively. Therefore, transfer matrices F_y and F_u also indicate the topology of the network. Since the target network is sparse, F_y and F_u are sparse matrices.

We divide the network into subsystems, each of which presents the regulation of a particular node from the others nodes and inputs. These subsystems are expressed in a non-parametric way using impulse responses so that system order of the transfer functions in F_y and F_u needs not be specified *a priori*. For the implementation reason, we truncate the impulse responses after sample time T as chapter 5. T is set sufficiently large to catch major dynamics of impulse responses which decay exponentially fast. For the ith node, we have:

(6.5.2)

$$y_i(t) = \sum_{j=1}^{p}\sum_{k=1}^{T} h_{ij}^{y}(k)y_j(t-k) + \sum_{j=1}^{m}\sum_{k=1}^{T} h_{ij}^{u}(k)u_j(t-k) + \hat{e}_i(t)$$

where h_{ij}^{y} and h_{ij}^{u} denote impulse responses of transfer functions $\left[F_y\right]_{ij}$ and $[F_u]_{ij}$, respectively. These impulse responses are assumed to be stable. Note that if a node or input does not control the ith node, its corresponding term in (6.5.2) is removed from the model.

Assume time series data from indices 1 to N for each node and input are collected for inference. We define the following matrices and vectors associated to the ith node:

(6.5.3)

$$Y = \begin{bmatrix} y_N^i \\ \vdots \\ y_{T+1}^i \end{bmatrix},\ w_k = \begin{bmatrix} w_{k1} \\ \vdots \\ w_{kf(k)} \end{bmatrix}$$
$$A_k = \left[\begin{array}{ccc|ccc} y_{N-1:N-T}^1 & \cdots & y_{N-1:N-T}^p & u_{N-1:N-T}^1 & \cdots & u_{N-1:N-T}^m \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y_{T:1}^1 & \cdots & y_{T:1}^p & u_{T:1}^1 & \cdots & u_{T:1}^m \end{array}\right]$$
$$\sigma^2 = E\{\hat{e}_i(t)^2\}$$

where $k \in \{1,2,3\ldots\}$ is the index for all possible zero structures of $[F_y(i,:)\quad F_u(i,:)]$, each of which corresponds to a particular topology. Therefore, we attain a finite collection of models $\{\mathcal{M}_k\}$ within which model structures are different from model to model. We use a Boolean vector F_k to represent the zero structure (i.e. network topology) of $[F_y(i,:)\quad F_u(i,:)]$ of $\mathcal{M}_k$. $f(k)$ maps index k to the number of non-zero elements in F_k. $w_k \in \mathbb{R}^{Tf(k)}$ contains all the truncated impulse responses of the non-zero entries of $[F_y(i,:)\quad F_u(i,:)]$ and block $w_{kj} \in \mathbb{R}^{T}$ denotes the impulse response of a node or input. $A_k \in \mathbb{R}^{T(N-T)\times Tf(k)}$ includes data of all the associated nodes and inputs that are controlling the ith node. $Y \in \mathbb{R}^{N-T}$ contains the data of the ith node. σ^2 is the noise variance of process noise. Superscripts denote the index of nodes and inputs. It should be noticed that the dimension of w_k and A_k varies with index k since the number of nodes and inputs that control the ith node changes with respect to topology F_k. Because the ground truth topology is unknown, index k also remains to be determined.

For the ith node, we have the likelihood function based on Bayes' rules:

$$p(y_{T+1:N}^{i}|w_k,\sigma^2,k)=(2\pi\sigma^2)^{-\frac{N-T}{2}}exp\left(-\frac{1}{2\sigma^2}\|Y-A_kw_k\|_2^2\right) \tag{6.5.3}$$

where the dependence on the initial conditions, nodes and inputs is suppressed for convenience.

It is then important to introduce prior distributions to promote network sparsity (related to the number of connectivity among observed nodes), model parsimony (related to the number of hidden nodes and their connectivity with observed nodes) and system stability to the proposed model.

6.6 Full Bayesian model for DSF

6.6.1 Prior Distribution

The kernel method discussed in chapter 5 treats network topology as a deterministic but unknown quantity. Network topology is reflected by estimated non-zero impulse responses. Nevertheless, this approach does not allow to evaluate inference confidence and is sensitive to local optimal solutions. In addition, the kernel method has difficulty dealing with the noise variance. In this chapter, we construct a full Bayesian model where network topology is regarded as a random quantity. We explore the full Bayesian model directly via numerical sampling rather than through its analytical approximation. Hence, we can assess the confidence of the inferred network and reduce the risk of being trapped at local optimal solutions. More importantly, numerical sampling offers a good estimation of process noise thus making the developed method more robust to noise perturbations.

Full Bayesian treatment requires introducing prior distributions for all the random variables including impulse responses w_k, noise variance σ^2 and topology index k. These priors reflect initial belief of the system as well as basic properties of these random quantities. In addition, parameters of priors (hyperparameter) are also treated as random variables and hyperpriors are introduced to these hyperparameters.

We follow the Bayesian framework of the kernel method to introduce priors for impulse responses. Impulse responses are assumed to be independent Gaussian processes like chapter 5:

$$p(w_{kq}|\beta_{kq},\lambda_{kq},k)\sim\mathcal{N}(w_{kq}|\mathbf{0},K_{kq}) \tag{6.6.1}$$

where

$$\begin{aligned}\left[K_{kq}\right]_{ij}&=k(i,j)\\ k(i,j)&=\lambda_{kq}k'(i,j;\beta_{kq})\\ k'(i,j;\beta)&=\frac{e^{-\beta(i+j)}e^{-\beta\max(i,j)}}{2}-\frac{e^{-3\beta\max(i,j)}}{6}\end{aligned}$$

$K_{kq}\in\mathbb{R}^{T\times T}$ is the covariance matrix of the qth group of w_k. $k'(i,j;\beta)$ is the discrete stable spline kernel controlled by hyperparameter $\beta\geq 0$ which determines the decaying rate

of impulse responses. The kernel function is also scaled by another hyperparameter, $\lambda \geq 0$ that is related to the amplitude of impulse responses [113]. The complete prior for w_k thus is:

(6.6.2)

$$p(w_k|\beta_k, \lambda_k, k) \sim \mathcal{N}(w_k|\mathbf{0}, K_k)$$

where $K_k \in \mathbb{R}^{Tf(k)\times Tf(k)}, \beta_k \in \mathbb{R}^{f(k)}$, $\lambda_k \in \mathbb{R}^{f(k)}$ and:

$$K_k = blkdiag\{K_{k1}, \dots, K_{kf(k)}\}$$
$$\beta_k = [\beta_{k1}, \dots, \beta_{kf(k)}]'$$
$$\lambda_k = [\lambda_{k1}, \dots, \lambda_{kf(k)}]'$$

Based on the likelihood function in (6.5.3), we use Inverse-Gamma distribution $IG(\sigma^2; a, b)$ as the conjugate prior for noise variance σ^2 [40], [90]. In the absence of a specific preference, a and b are set extremely small (0.001) to make the prior non-informative [180]:

(6.6.3)

$$p(\sigma^2; a, b) = \frac{b^a}{\Gamma(a)}(\sigma^2)^{-a-1}e^{-\frac{b}{\sigma^2}}$$

The index, k is a positive integer and no bigger than $k_{max} = \sum_{i=0}^{p+m-1} \boldsymbol{C}(p+m-1, i)$ that equates to the total number of all possible topologies of (6.5.3), where $\boldsymbol{C}$ denotes the combination operator: $\boldsymbol{C}(n, m) = \frac{n(n-1)\cdots(n-m+1)}{m!}$. Its prior is the variant of a truncated Poisson distribution with rate parameter α [174], [180]:

(6.6.4)

$$p(k|\alpha) = \frac{\alpha^{f(k)}/f(k)!}{\sum_{j=1}^{k_{max}} \alpha^{f(j)}/f(j)!}$$

It can be seen that topology indices which are mapped to the same number of links by $f(k)$ possess the same probability even if their corresponding zero structures are different.

Finally, there are three hyperparameters left. For λ_k, we adopt Inverse Gamma distribution $p(\lambda_k; e, f)$ as the hyperprior. This distribution is the conjugate prior of λ_k as it is the scale parameter of a Gaussian distribution. We set $e = 2$ and $f = 1$ so that the distribution has infinite variance but also puts most of weights over small values. Note that λ_k also serves as an ARD parameter to promote sparsity. Under the framework of kernel methods, it is determined by solving an optimization problem. In this case, the power of ARD mechanisms is limited by the difficulty of finding the optimal solution. In this chapter, we apply a sampling method (i.e. RJMCMC) to deliberately set certain elements of λ_k zero in different transition moves so that the effect of ARD is maximally activated. We use the same hyperprior $p(\beta_k; c, d)$ (i.e. Inverse Gamma distribution) for non-negative hyperparameter β_k. Hyperparameter β_k controls the decaying rate of impulse responses. We find that its value, if bigger than 10, leads to fast decaying impulses responses that represent negligible system dynamics. As a result, we set $c = 2$ and $d = 1$ so that the prior concentrates on the region, $(0,10]$ but still has infinite variance to support the other regions of the sample space.

(6.6.5)

$$p(\beta_k; c, d) = \prod_{i=1}^{f(k)} IG(\beta_{ki}; c, d)$$

$$p(\lambda_k; e, f) = \prod_{i=1}^{f(k)} IG(\lambda_{ki}; e, f)$$

A conjugate Gamma distribution, $Gamma(\alpha; \delta_1, \delta_2)$ for the rate parameter of a Poisson distribution is assigned to be the prior for α [174], [180]. As α is the mean of a Poisson distribution, we set $\delta_1 = 0.1$ and $\delta_2 = 1$ to reflect the belief of sparse topology ($f(k) \ll p$). Nevertheless, we will see that this setting barely influences the final posterior distribution.

(6.6.6)

$$p(\alpha) = \frac{{\delta_2}^{\delta_1}}{\Gamma(\delta_1)} \alpha^{\delta_1 - 1} e^{-\delta_2 \alpha}$$

6.6.2 Posterior Distribution

Based on Bayes' rules, we have the posterior distribution of DSFs as:

(6.6.7)

$$\begin{aligned}
&p(k, w_k, \beta_k, \lambda_k, \sigma^2, \alpha | Y) \\
&\propto p(Y | w_k, \sigma^2) p(w_k | \beta_k, \lambda_k, \sigma^2) p(\sigma^2) p(\beta_k | k) p(\lambda_k | k) p(k | \alpha) p(\alpha) \\
&\propto (2\pi\sigma^2)^{-\frac{N-T}{2}} exp\left(-\frac{1}{2\sigma^2} \|Y - A_k w_k\|_2^2\right) \\
&\times \prod_i^{f(k)} |2\pi K_{ki}|^{-\frac{1}{2}} exp\ (-\frac{1}{2} w'_{ki} K_{ki}^{-1} w_{ki}) \\
&\times (\sigma^2)^{-a-1} exp\ (-\frac{b}{\sigma^2}) \\
&\times \prod_i^{f(k)} IG(\beta_{ki}; c, d) IG(\lambda_{ki}; e, f) \times \frac{\alpha^{f(k)}/f(k)!}{\sum_{j=1}^{k_{max}} \alpha^{f(j)}/f(j)!} \times \alpha^{\delta_1 - 1} e^{-\delta_2 \alpha}
\end{aligned}$$

where subscript k denotes that the quantity dimension depends on index k. The dependence on initial conditions, other nodes and inputs is suppressed for convenience.

By completing squares, (6.6.7) becomes:

(6.6.8)

$$\begin{aligned}
&p(k, w_k, \beta_k, \lambda_k, \sigma^2, \alpha | Y) \\
&\propto p(Y | w_k, \sigma^2) p(w_k | \beta_k, \lambda_k, \sigma^2) p(\sigma^2) p(\beta_k | k) p(\lambda_k | k) p(k | \alpha) p(\alpha) \\
&\propto (2\pi\sigma^2)^{-\frac{N-T}{2}} exp\left\{-\frac{1}{2} Y'(\sigma^2 I + A_k K_k A'_k)^{-1} Y\right\} \\
&\times |2\pi K_k|^{-\frac{1}{2}} exp\ \left\{-\frac{1}{2}(w_k - \mu_k)' \Sigma_k^{-1} (w_k - \mu_k)\right\} \\
&\times (\sigma^2)^{-a-1} exp\ (-\frac{b}{\sigma^2})
\end{aligned}$$

$$\times \prod_{i}^{f(k)} IG(\beta_{ki}; c, d) IG(\lambda_{ki}; e, f) \times \frac{\alpha^{f(k)}/f(k)!}{\sum_{j=1}^{k_{max}} \alpha^{f(j)}/f(j)!} \times \alpha^{\delta_1 - 1} e^{-\delta_2 \alpha}$$

where

$$\Sigma_k^{-1} = \frac{1}{\sigma^2} A'_k A_k + K_k^{-1}$$

$$\mu_k = \frac{1}{\sigma^2} \Sigma_k A'_k Y$$

Based on the full posterior distribution in (6.6.8), we can easily extract the conditional posterior distributions of w_k, σ^2 and α:

(6.6.9)

$$p(w_k|\beta_k, \lambda_k, k, \sigma^2, \alpha, Y) \sim \mathcal{N}(w_k|\mu_k, \Sigma_k)$$

$$p(\sigma^2|w_k, \beta_k, \lambda_k, k, \alpha, Y) \sim IG(\sigma^2; a + \frac{N-T}{2}, \frac{1}{2}\|Y - A_k w_k\|_2^2 + b)$$

$$p(\alpha|w_k, \beta_k, \lambda_k, k, \sigma^2, Y) \propto \frac{\alpha^{\delta_1 - 1 + f(k)}/f(k)!}{\sum_{j=1}^{k_{max}} \alpha^{f(j)}/f(j)!} e^{-\delta_2 \alpha}$$

In addition, by marginalizing w_k out from the full distribution in (6.6.8), we get the reduced joint posterior distribution of index k and hyperparameters β_k and λ_k:

(6.6.10)

$$p(k, \beta_k, \lambda_k|\sigma^2, \alpha, Y)$$

$$\propto exp\left\{-\frac{1}{2} Y'(\sigma^2 I + A_k K_k A'_k)^{-1} Y\right\} \times |\sigma^2 I + A_k K_k A'_k|^{-\frac{1}{2}}$$

$$\times \prod_{i}^{f(k)} IG(\beta_{ki}; c, d) IG(\lambda_{ki}; e, f) \times \frac{\alpha^{f(k)}/f(k)!}{\sum_{j=1}^{k_{max}} \alpha^{f(j)}/f(j)!}$$

This marginal distribution will be used later to form PCG samplers.

Remark 6.6.1: If the model contains measurement noise, transfer matrices F_y and F_u are full (section 5.11). In this case, the dimension of random variables is fixed. As a result, a normal MH algorithm can be used to draw samples from the full Bayesian model. Nevertheless, hyperparameter λ_k is no longer an ARD parameter. The Bayesian model cannot promote network sparsity. We will address this issue later in chapter 9.

6.7 Sampling full Bayesian model with fixed topology

With the full Bayesian model, we are interested in the posterior probability of network topology, $p(k|Y)$ by which we can evaluate the probability of different topologies given measured data. We can also estimate the most likely network topology by solving the MAP problem, $arg\max_k p(k|Y)$. Given the estimated k, we can estimate impulse responses and

noise variance by computing their expectation, $E(w_k|k,Y)$ and $E(\sigma^2|k,Y)$, which requires to evaluate $p(w_k|k,Y)$ and $p(\sigma^2|k,Y)$. However, analytical expressions for $p(k|Y)$, $p(w_k|k,Y)$ and $p(\sigma^2|k,Y)$ are intractable since we have to perform high-dimensional integrals of the nonlinear Bayesian model in (6.6.7) [180]. To solve the problem, this chapter applies numerical sampling methods. The true distributions can be accurately approximated by the empirical ones built using the samples drawn from the full Bayesian model. A MCMC technique is applied in this section to sample the probabilistic model in (6.6.7). Under some mild conditions, the empirical distribution will converge to the invariant distribution of the Markov chain with probability 1 [165]. The ability of this sampling framework to evaluate the uncertainty of both impulse responses and network topology is highly attractive compared with deterministic approximation techniques.

To begin with, we assume the topology of the target network is known *a priori* so that k is fixed. In this case, a traditional MCMC method is capable of sampling the distribution, $p(w_k,\beta_k,\lambda_k,\sigma^2,\alpha|k,Y)$ since the dimension of random variables is unchanged. We marginalize out some random quantities during sampling to improve the convergence property. We apply the techniques (marginalization, permutation and trimming) discussed in section 6.2 to deduce a MH-within-PCG sampler for $p(w_k,\beta_k,\lambda_k,\sigma^2,\alpha|k,Y)$. The designed sampler is modified later to sample the full Bayesian model in (6.6.7) where the network topology is unknown and treated as a random variable.

We first propose a normal Gibbs sampler within which the sampling of β_k and λ_k is blocked together in Sampler 1.

Sampler 1 Block Gibbs sampler
1: Sample $p(w_k^{t+1}|\beta_k^t,\lambda_k^t,(\sigma^2)^t,\alpha^t,k,Y)$
2: Sample $p(\beta_k^{t+1},\lambda_k^{t+1}|w_k^{t+1},(\sigma^2)^t,\alpha^t,k,Y)$
3: Sample $p((\sigma^2)^{t+1}|w_k^{t+1},\beta_k^{t+1},\lambda_k^{t+1},\alpha^t,k,Y)$
4: Sample $p(\alpha^{t+1}|w_k^{t+1},\beta_k^{t+1},\lambda_k^{t+1},(\sigma^2)^{t+1},k,Y)$

Steps 2 and 4 in Sampler 1 must be replaced by MH schemes since the distributions of these steps cannot be sampled directly. Distributions of these two steps are only known up to a constant according to section 6.6. By replacing steps 2 and 4 with MH algorithms, we achieve a MH-within-Gibbs sampler in Sampler 2.

Sampler 2 MH-within-Gibbs sampler
1: Sample $p(w_k^{t+1}|\beta_k^t,\lambda_k^t,(\sigma^2)^t,\alpha^t,k,Y)$
2: Sample $(\beta_k^{t+1},\lambda_k^{t+1})$ using Metropolis-Hastings with $p(\beta_k,\lambda_k|w_k,\sigma^2,\alpha,k,Y)$ as the invariant distribution
3: Sample $p((\sigma^2)^{t+1}|w_k^{t+1},\beta_k^{t+1},\lambda_k^{t+1},\alpha^t,k,Y)$
4: Sample α^{t+1} using Metropolis-Hastings with $p(\alpha|\beta_k,\lambda_k,w_k,\sigma^2,k,Y)$ as the invariant distribution

Sampler 2 is proper because the replacement maintains the invariant distributions of the original steps. The construction of this hybrid sampler is based on the direct use of the composition operation of transition kernels.

We then apply the rule of marginalization to Sampler 2 in order to reduce the number of variables being conditioned on. Variable w_k is marginalized out from step 2 leading to Sampler 3. The reduced step is then followed immediately by a direct draw from the full conditional distribution of w_k. A quantity is labelled with an asterisk if it is not conditioned on in the next step.

Sampler 3 Marginalization
1: Sample $p(w_k^\star|\beta_k^t, \lambda_k^t, (\sigma^2)^t, \alpha^t, k, Y)$
2: Sample $(\beta_k^{t+1}, \lambda_k^{t+1})$ using Metropolis-Hastings with $p(\beta_k, \lambda_k|\sigma^2, \alpha, k, Y)$ as the invariant distribution
3: Sample $p(w_k^{t+1}|\beta_k^{t+1}, \lambda_k^{t+1}, (\sigma^2)^t, \alpha^t, k, Y)$
4: Sample $p((\sigma^2)^{t+1}|w_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, \alpha^t, k, Y)$
5: Sample α^{t+1} using Metropolis-Hastings with $p(\alpha|\beta_k, \lambda_k, w_k, \sigma^2, k, Y)$ as the invariant distribution

Before we proceed, we give a simple proof to show why the marginalization is valid.

Theorem 6.7.1 *(Marginalization of MH-within-PCG):*
The invariant distribution of an MH step is preserved if this step is replaced by a reduced MH step followed immediately by a direct draw from the complete conditional distribution of that reduced quantity.

Proof:
Suppose $p^\star(x, y)$ is the invariant distribution of a full MH step. After marginalizing y, the sampler becomes a reduced MH sampling from $p^\star(x)$ followed by a direct sampling from $p^\star(y|x)$. Assume $k(x, x')$ is the density of the transition kernel of the reduce MH step. The marginal distribution of Markov state is:

$$\begin{aligned} p_{t+1}(x', y') &= \int p_t(x, y)p(x', y'|x, y)dxdy \\ &= \int p^\star(x, y)k(x, x')p^\star(y'|x')dxdy \\ &= p^\star(y'|x')\int p^\star(x)k(x, x')dx \end{aligned}$$

Since $p^\star(x') = \int p^\star(x)k(x, x')dx$

$$p_{t+1}(x', y') = p^\star(y'|x')p^\star(x') = p^\star(x, y)$$

According to Theorem 6.7.1, the invariant distribution is preserved if two sampling steps (reduced MH and direct sampling) are executed contiguously. That means when applying permutation to the sampler, these two steps cannot be separated or reordered.

w_k is sampled twice in Sampler 3. Its first sample, $w_k^\star$ in step 1 is redundant because it is not conditioned on in the next step and is covered by the sampling in step 3. Since there is no need to perform permutation here, we can trim out the first step directly. After trimming, we achieve a sampling scheme described in Sampler 4.

Note that the valid sequence of steps in Sampler 4 is not unique. The arrangement of steps 3 and 4 is quite flexible. For example, Sampler 4 can be rearranged as $4 \to 3 \to 1 \to 2$.

However, not all the permutations are valid. For instance, sequence $2 \to 1 \to 3 \to 4$ (modified Sampler 4) derived from trimming step 3 of Sampler 3 is incorrect because the procedure violates the rule of trimming. The direct sampling from $p(w_k|\beta_k, \lambda_k, \sigma^2, \alpha, k, Y)$ after marginalization cannot be dropped because w_k is conditioned on in the next step. As a result, the sampler $2 \to 1 \to 3 \to 4$ no longer preserves the original invariant distribution. Unfortunately, this issue was not considered carefully in much of the previous research [210].

Sampler 4 Permutation and Trimming
1: Sample $(\beta_k^{t+1}, \lambda_k^{t+1})$ using Metropolis-Hastings with $p(\beta_k, \lambda_k|\sigma^2, \alpha, k, Y)$ as the invariant distribution
2: Sample $p(w_k^{t+1}|\beta_k^{t+1}, \lambda_k^{t+1}, (\sigma^2)^t, \alpha^t, k, Y)$
3: Sample $p((\sigma^2)^{t+1}|w_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, \alpha^t, k, Y)$
4: Sample α^{t+1} using Metropolis-Hastings with $p(\alpha|\beta_k, \lambda_k, w_k, \sigma^2, k, Y)$ as the invariant distribution

The sampled distributions of each step in Sampler 4 have already been discussed in section 6.6. Sampling steps 2 and 3 are trivial since their target distributions are Inverse-Gamma and Gaussian. We only need to design algorithms for steps 1 and 4. For the MH approach, we first have to design a proposal distribution for Markov states to produce candidate samples and then deduce the corresponding acceptance probability.

Since β_k and λ_k are non-negative, we use truncated normal distributions centered at the current states β_k^t or λ_k^t with small variance (0.02) as the proposal distributions. Small variance is adopted in order to avoid the high rejection rate of proposals but at the cost of lower convergence speed. This variance is tuned during inference to achieve an acceptance rate around 35% according to a heuristic rule in [220]. The newly proposed β_k^{prop} and λ_k^{prop} can be treated as the perturbed version of the current state. For a Gaussian distribution with mean μ and variance σ, its truncated probability density function on $[a, b]$ is:

(6.7.1)

$$p(x; \mu, \sigma, a, b) = \frac{\phi\left(\frac{x-\mu}{\sigma}\right)}{\sigma\left[\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)\right]}$$

where

$$\phi(x) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{1}{2}x^2\right)$$

$$\Phi(x) = \frac{1}{2}\left[1 + erf\left(\frac{x}{\sqrt{2}}\right)\right]$$

$$erf(x) = \frac{1}{\sqrt{\pi}}\int_{-x}^{x} e^{-t^2} dt$$

According to the feasible domain of β_k and λ_k, $a = 0$ and $b = +\infty$.

To guarantee the invariant distribution of the Markov chain is $p(\beta_k, \lambda_k|\sigma^2, \alpha, k, Y)$, a sufficient condition called 'detailed balance' must be met. In what follows, the variables other than β and λ are suppressed for simplicity:

(6.7.2)

$$p(\beta^t, \lambda^t) pr(\beta^{prop}, \lambda^{prop} | \beta^t, \lambda^t) A(\beta^{prop}, \lambda^{prop} | \beta^t, \lambda^t)$$
$$= p(\beta^{prop}, \lambda^{prop}) pr(\beta^t, \lambda^t | \beta^{prop}, \lambda^{prop}) A(\beta^t, \lambda^t | \beta^{prop}, \lambda^{prop})$$

where superscript $prop$ denotes the proposed quantity, pr represents the proposal distribution and function $A(\cdot)$ denotes the acceptance probability.

The acceptance probability based on (6.7.2) is:

$$A(\beta^{prop}, \lambda^{prop} | \beta^t, \lambda^t) = \min\left\{1, \quad \frac{p(\beta^{prop}, \lambda^{prop}) pr(\beta^t, \lambda^t | \beta^{prop}, \lambda^{prop})}{p(\beta^t, \lambda^t) pr(\beta^{prop}, \lambda^{prop} | \beta^t, \lambda^t)}\right\} \tag{6.7.3}$$

The MH algorithm to sample α for step 4 in Sampler 4 is designed in the same way. We use a Gamma distribution as the proposal distribution for $\alpha > 0$ [174]:

$$pr(\alpha) \propto \alpha^{\delta_1 - 1 + f(k)} e^{-\alpha(1+\delta_2)} \tag{6.7.4}$$

so that

$$r(\alpha^{prop} | \alpha) = \frac{p(\alpha^{prop}) pr(\alpha | \alpha^{prop})}{p(\alpha) pr(\alpha^{prop} | \alpha)}$$
$$= \frac{e^{-\alpha} \sum_{j=1}^{k_{max}} (\alpha)^{f(j)} / f(j)!}{e^{-\alpha^{prop}} \sum_{j=1}^{k_{max}} (\alpha^{prop})^{f(j)} / f(j)!}$$
$$A(\alpha^{prop} | \alpha) = \min\{1, \quad r(\alpha^{prop} | \alpha)\}$$

Note that step 4 and other sampling steps in Sampler 4 are independent. That is because hyperparameter α is only related to random variable k that is conditioned on by all random quantities. As a result, removing step 4 from the sampler will not affect the sampling of other random variables. However, if we sample k as well (in the following sections), sampling α is inevitable. For this reason, we prefer to keep the sampling step of α in Sampler 4.

6.8 Assembly of reversible jump MCMC and MH-within-PCG

If index k is not fixed (i.e. the network topology is unknown), we treat k as a random variable and draw its samples as well from the full Bayesian model in (6.6.8). To do that, we first block β_k, λ_k and k together in Sampler 5 where the joint posterior distribution is given in (6.6.10).

Sampler 5 Block Gibbs sampler
1: Sample $p(w_k^{t+1} | \beta_k^t, \lambda_k^t, (\sigma^2)^t, \alpha^t, k^t, Y)$
2: Sample $p(\beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1} | w_k^{t+1}, (\sigma^2)^t, \alpha^t, Y)$
3: Sample $p((\sigma^2)^{t+1} | w_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, \alpha^t, k^{t+1}, Y)$
4: Sample $p(\alpha^{t+1} | w_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, (\sigma^2)^{t+1}, k^{t+1}, Y)$

Although index k is sampled in step 2 of Sampler 5, we find that $k^{t+1} = k^t$ in each iteration due to the sampled w_k in step 1 whose structure determines network topology. Therefore, the model class is fixed to the initial $\mathcal{M}_{k^0}$. The posterior distribution is not explored

sufficiently. To solve this problem, we have to marginalize out variable w_k in step 2. According to the rule of marginalization, after marginalizing variable w_k out from step 2, w_k must be sampled immediately from its full conditional distribution, thus leading to Sampler 6.

Sampler 6 Marginalization
1: Sample $p(w_k^\star|\beta_k^t, \lambda_k^t, (\sigma^2)^t, \alpha^t, k^t, Y)$
2: Sample $(\beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1})$ using Metropolis-Hastings with $p(\beta_k, \lambda_k, k|\sigma^2, \alpha, Y)$ as the invariant distribution
3: Sample $p(w_k^{t+1}|\beta_k^{t+1}, \lambda_k^{t+1}, (\sigma^2)^t, \alpha^t, k^{t+1}, Y)$
4: Sample $p((\sigma^2)^{t+1}|w_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, \alpha^t, k^{t+1}, Y)$
5: Sample α^{t+1} using Metropolis-Hastings with $p(\alpha|\beta_k, \lambda_k, w_k, \sigma^2, k, Y)$ as the invariant distribution

Step 1 of Sampler 6 is redundant because $w_k^\star$ is not conditioned on in step 2. Therefore, step 1 can be trimmed out from the sampler. With the first step discarded, we have the final sampler in Sampler 7:

Sampler 7 MH-within-PCG
1: Sample $(\beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1})$ using Metropolis-Hastings with $p(\beta_k, \lambda_k, k|\sigma^2, \alpha, Y)$ as the invariant distribution
2: Sample $p(w_k^{t+1}|\beta_k^{t+1}, \lambda_k^{t+1}, (\sigma^2)^t, \alpha^t, k^{t+1}, Y)$
3: Sample $p((\sigma^2)^{t+1}|w_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, \alpha^t, k^{t+1}, Y)$
4: Sample α^{t+1} using Metropolis-Hastings with $p(\alpha|\beta_k, \lambda_k, w_k, \sigma^2, k, Y)$ as the invariant distribution

Steps 2, 3 and 4 have already been discussed in section 6.7. Since index k is conditioned on in these steps, the dimension of the sampled variables is fixed. Hence, Sampler 4 can be used directly for these steps. It can be seen that Sampler 7 can explore different model classes (related to differing topologies) $\mathcal{M}_k$ if step 1 is well designed. Since the dimension of the random variables in step 1 varies with model classes, a normal MCMC method is incapable of drawing samples from the target distribution. Therefore, we resort to RJMCMC schemes.

The basic idea to draw samples from the reduced marginal distribution, $p(\beta_k, \lambda_k, k|\sigma^2, \alpha, Y)$ is to design different moves for the Markov chain to traverse between different parameter subspaces [126]. The proposed moves must ensure that it is possible to visit all parameter subspaces so that the entire sample space is sufficiently explored. For this purpose, we come up with three types of moves.

Birth move:

The number of links in the next state indicated by $f(k^{t+1})$ is one more than that of the current state (i.e. $f(k^{t+1}) = f(k^t) + 1$). Furthermore, the zero structure of $\mathcal{M}_{k^t}$ and $\mathcal{M}_{k^{t+1}}$ only differs at one entry. For example, F_{k^t} is $[1 \quad 0 \quad 0]$ and $F_{k^{t+1}}$ is $[1 \quad 0 \quad 1]$.

Death move:

The number of links of the next state indicated by $f(k^{t+1})$ is one less than that of the current state (i.e. $f(k^{t+1}) = f(k^t) - 1$). Moreover, the zero structure of $\mathcal{M}_{k^t}$ and $\mathcal{M}_{k^{t+1}}$

only differs at one entry. For example, F_{k^t} is $[1 \quad 0 \quad 1]$ and $F_{k^{t+1}}$ is $[1 \quad 0 \quad 0]$.

Update move:

The topology is unchanged in the next state ($k^{t+1} = k^t$) but other random variables are updated.

The attempt to add an extra node (birth move) into the model and remove one node from the model (death move) naturally encourages a global search of the parameter subspaces [180]. The death move is equivalent to setting an ARD parameter λ_{ki} zero directly whereas the birth move reverses this process and retrieves non-zero λ_{ki}. As a result, the effect of ARD is maximally activated. The update move inherently infers internal dynamics of the network given the fixed topology.

Network topology is represented by the dimensionality of different parameter subspaces while model parameters are points contained in these subspaces. Inference methods applying deterministic approximations (like the kernel method and SBL) search for the optimal solution in a unique parameter space of $\mathbb{R}^{(p+m)T}$ (including all the nodes and inputs), where network topology is determined by the zero structure of the solution. The local optimal solutions whose zero structure is not consistent with the true topology are equally likely to be picked up by the algorithm, depending on the initial point. Although a heuristic search of the optimal solution can be conducted by running the algorithm with different initial points, this scheme is very inefficient and is prohibitive for high-dimensional parameter spaces. That is mainly because the estimation of network topology and model parameters is blended together. In contrast, the RJMCMC scheme decomposes the inference task through three types of moves. Topology detection is performed via jumps between parameter subspaces while parameter estimation is conducted via Monte Carlo sampling. Therefore, RJMCMC provides a more efficient way to conduct a randomized global search of the entire sample space. RJMCMC can effectively avoid all the local maxima in some parameter subspaces and favor the subspaces whose corresponding topologies are similar to the ground truth by paying more frequent visits. Although samples may surround a suboptimal point of model parameters representing biased internal dynamics (local maxima of the distribution), the topology indicated by that parameter subspace can still be correct.

To realize 'Birth' and 'Death' moves, we design the procedure in Algorithm 1 and 2.

Algorithm 1 Birth (Add 1 link)

1: With probability P_B, choose Birth move.

2: Select a node to be added to the current topology randomly by Uniform distribution

$$q_B(j|k^t) = \frac{1}{p + m - f(k^t)}$$

3: Propose β^{prop} and λ^{prop} of the jth node by sampling $pr_B(\beta, \lambda) = pr_B(\beta)pr_B(\lambda)$ with the hyperparameters of other nodes unchanged. Insert β^{prop} and λ^{prop} to β^t and λ^t to generate β^{t+1} and λ^{t+1}.

4. Accept with probability A_B.

In Algorithm 1, we adopt $pr_B(\beta) = IG(\beta; c, d)$ and $pr_B(\lambda) = IG(\lambda; e, f)$ as the proposal distributions for β and λ, respectively, which are the same with their prior distributions. By

doing so, the calculation of the acceptance probability is simplified.

Algorithm 2 Death (Remove 1 link)
1: With probability P_D, choose Death move.
2: Select a node to be removed from the current topology randomly by Uniform distribution

$$q_D(j|k^t) = \frac{1}{f(k^t) - 1}$$

3: Remove β and λ of the jth node from β^t and λ^t with other elements unchanged to generate β^{t+1} and λ^{t+1}.
4. Accept with probability A_D.

The acceptance probability for birth and death moves is calculated based on 'detailed balance':

(6.8.1)

$$p(\beta_k^t, \lambda_k^t, k^t) P_B q_B(j|k^t) pr_B(\beta^{prop}, \lambda^{prop}) A_B = p\left(\beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1}\right) P_D q_D(j|k^{t+1}) A_D$$

where the Jacobian of the transformation map is the identity matrix.

As a result, the acceptance probability is:

(6.8.2)

$$r_B(\beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1} | \beta_k^t, \lambda_k^t, k^t)$$
$$= \frac{P_D}{P_B} \frac{exp\left\{-\frac{1}{2} Y'\left(\sigma^2 I + A_k K_k^{t+1} A'_k\right)^{-1} Y\right\}}{exp\left\{-\frac{1}{2} Y'(\sigma^2 I + A_k K_k^t A'_k)^{-1} Y\right\}} \frac{\left|\sigma^2 I + A_k K_k^{t+1} A'_k\right|^{-\frac{1}{2}}}{\left|\sigma^2 I + A_k K_k^t A'_k\right|^{-\frac{1}{2}}} \frac{\alpha[p + m - f(k^t)]}{f(k^{t+1})[f(k^{t+1}) - 1]}$$

$$r_D\left(\beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1} \middle| \beta_k^t, \lambda_k^t, k^t\right) = r_B^{-1}(\beta_k^t, \lambda_k^t, k^t | \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1})$$
$$A_B = \min\{1, \quad r_B\}$$
$$A_D = \min\{1, \quad r_D\}$$

Finally, the update move is shown in Algorithm 3. Since the topology is fixed, the proposal distributions and acceptance probability are exactly the same with Sampler 4.

Algorithm 3 Update (Retain the current topology)
1: With probability P_U, choose Update move.
2: Propose β_k^{prop} and λ_k^{prop} using truncated normal distributions.
3. Accept with probability A_U.

Note that the probability of three moves (P_B, P_D and P_U) depends on k^t. We set $P_B = P_D = 0.3$ if $1 < f(k^t) < n + p$. $P_B = 0$, $P_D = 0.6$ if $f(k^t) = p + m$ and $P_D = 0$, $P_B = 0.6$ if $f(k^t) = 1$. In addition, $P_U = 1 - P_B - P_D$ in all cases.

To conclude, we have the final Algorithm 4.

Algorithm 4 Sampler for the full Bayesian model
1: ***Initialization***: set $\left(k^0, \beta_k^0, \lambda_k^0, w_k^0, (\sigma^2)^0, \alpha^0\right)$
2: ***For*** $t = 1{:}\,MAX$ do
3: Sample $u \sim \mathcal{U}[0,1]$ (Uniform distribution)

4: ***If*** $u \leq P_B$ then
5: Birth move generating $(\beta_k^t, \lambda_k^t, k^t)$
6: ***else if*** $u \leq P_B + P_D$ then
7: Death move generating $(\beta_k^t, \lambda_k^t, k^t)$
8: ***else***
9: Update move generating $(\beta_k^t, \lambda_k^t, k^t)$
10: ***end if***
11: Sample w_k^t from $p(w_k^t|\beta_k^t, \lambda_k^t, k^t, (\sigma^2)^{t-1}, \alpha^{t-1}, Y)$
12: Sample $(\sigma^2)^t$ from $p((\sigma^2)^t|w_k^t, \beta_k^t, \lambda_k^t, k^t, \alpha^{t-1}, Y)$
13: Sample α^t using Metropolis-Hastings with with $p(\alpha|\beta_k, \lambda_k, w_k, \sigma^2, k, Y)$ as the invariant distribution
14: ***End for***

With the samples $\{X_t, t = 1,2,3 \dots, t_{max}\}$, we can establish the empirical posterior distribution of k as [165]:

(6.8.3)

$$\hat{p}(\mathcal{M}_j|Y) = \hat{p}(k = j|Y) = \frac{1}{t_{max}} \sum_{t=1}^{t_{max}} \mathbb{1}_{k^t}(j)$$

where

$$\mathbb{1}_x(j) = \begin{cases} 1 & \text{if } j = x \\ 0 & \text{if } j \neq x \end{cases}$$

The network topology is estimated as $\hat{k} = \max_k \hat{p}(k|Y)$. More importantly, the empirical distribution in (6.8.3) reveals the probability of all possible topologies, which indicates the confidence of the estimated $\hat{k}$ over the others.

Furthermore, impulses responses and noise variance can be evaluated as the mean of empirical distributions. If $\hat{k} = j$, they are estimated as:

(6.8.4)

$$\widehat{w_k} = \hat{E}(w_k|k = j, Y) = \frac{\sum_{t=1}^{t_{max}} w_k^t \mathbb{1}_j(k^t)}{\sum_{t=1}^{t_{max}} \mathbb{1}_j(k^t)}$$

$$\widehat{\sigma^2} = \hat{E}(\sigma^2|k = j, Y) = \frac{\sum_{t=1}^{t_{max}} (\sigma^2)^t \mathbb{1}_j(k^t)}{\sum_{t=1}^{t_{max}} \mathbb{1}_j(k^t)}$$

6.9 Computational cost of RJMCMC

The computational cost of the RJMCMC approach is very high due to two aspects. First of all, the computational cost of each iteration is expensive. Second, the algorithm is implemented in an iterative way so that the cost grows linearly with the total number of iterations. Hence, the computational burden can be prohibitive for large-scale networks.

The computational cost mainly comes from calculating the inversion and determinant of the

full covariance matrix, $C = \sigma^2 I + A_k K_k^t A'_k$. These two operations are executed twice in each iteration. They are first calculated in one of the three moves (using the proposed hyperparameters) in order to estimate the acceptance ratio. Following that, they are updated after sampling noise variance σ^2 for the use of the next iteration. The dimension of C is $\mathbb{R}^{(N-T)\times(N-T)}$, where N is the length of data, T is the length of truncated impulse responses and $N \gg T$. The matrix dimension is independent on the scale of the network but dominated by the length of data. Calculating matrix C requires $O((p+m)TN^2)$ work while calculating the inversion or determinant of C requires $O(N^3)$ work using standard methods. Hence, in total, calculating C^{-1} or $|C|$ demands $O(N^3 + (p+m)TN^2)$ work. Assuming rich data are available for inference (i.e. $N \gg (p+m)T$), the growth of the computational cost is cubic. As the Markov chain normally takes thousands of iterations to converge, the algorithm may take days to complete the inference (since the RJMCMC algorithm is implemented for each node independently). Therefore, finding a more effective and economical way to manipulate matrix C is critical.

Plenty of techniques have been developed to accelerate these operations such as greedy approximations, sub-sampling and Nystrom method. Nevertheless, most of these methods are ill-suited for calculating determinants. One of the state-of-the-art methods applies matrix factorization of hierarchical off-diagonal low-rank (HODLR) matrices [221], [222]. This method calculates matrix inversion and determinant at the cost of $O(n\log^2(n))$. The authors claim that their method works for most kernel functions and is simple to use. Nevertheless, the method may not scale well for some specific kernels such as those containing oscillation.

6.10 Simulation

Monte Carlo simulations were conducted to compare different approaches including GESBL, the kernel method and RJMCMC. To have a thorough investigation of the algorithm performance under different conditions, factors involving model classes, sparsity of network topologies, noise level and size of datasets were considered.

The accuracy of the inferred networks is evaluated using the average of True Positive Rate (TPR) and the average of Precision (Prec). TPR reveals the percentage of how many true links of the ground truth networks are identified, which implies the information richness of inference results. Prec indicates the accuracy of estimation. For example, if Prec is 50%, it means that half of the links in the estimated network are wrong and the inference result is not reliable.

6.10.1 Identify Multivariable ARX models

We first tested different methods on multivariable ARX models. The model structure is fairly simple and system dynamics can be characterized by finite impulse responses (FIR). We also considered networks possessing a ring structure in Figure 6.10.1. This class of networks is very sparse and contains a feedback loop, which should be more challenging to infer. SBL and GSBL were also applied to highlight the improvement of GESBL.

Monte Carlo simulations were conducted under two different conditions as follows.

Condition 1: *(Random sparse networks, Fixed number of data points, Different noise levels)*

The purpose of this simulation is to compare different inference methods and investigate how noise level influences their performance.

100 networks were generated with random topologies. The order and coefficients of polynomials were also randomized to cover a sufficiently rich class of ARX models. The maximum order of each entry of polynomial matrices, $A(z^{-1})$ and $B(z^{-1})$ was 5. Each node was independently driven by an input leading to a diagonal $B(z^{-1})$.

To be specific, a 10×10 sparse polynomial matrix $A(z^{-1})$ was first generated. Sparsity of the matrix relied on a predefined variable that controlled the probability for $[A(z^{-1})]_{ij}$ to be non-zero. Each polynomial entry of $A(z^{-1})$ was yielded by function $drmodel$ in Matlab so that all the roots were constrained within the unit circle of the complex plane. (Zeros of $A(z^{-1})$ are poles of $A^{-1}(z^{-1})$). The polynomial order was uniformly selected from 1 to 5 for each matrix entry. The stability of $A^{-1}(z^{-1})$ was then checked by calculating its poles using function $pole$ in Matlab. The matrix was discarded if it was unstable (i.e. any pole outside the unite circle of the complex plane). The parameters of polynomial matrix $B(z^{-1})$ were then generated using function $randn$ in Matlab as $B(z^{-1})$ had no impact on the system stability. The polynomial order of $B(z^{-1})$ was selected in the same way as $A(z^{-1})$. The above procedure was repeated until 100 stable models were generated.

On average, there were about 36 links in each network (out of 90 possible). Both the exciting input (known) and process noise (unknown) were independent Gaussian. The models were simulated with different Signal-Noise Ratio (SNR) and 100 time series data points for each node were collected. Here, SNR is defined as $SNR = 10\log_{10}\frac{\sigma_u^2}{\sigma_e^2}$ where σ^2 denotes signal variance. Input variance σ_u^2 was fixed to 1. The system order, k of the proposed model, $\mathcal{M}(w)$ for inference was set to 8 so that the generated ground truth model, $\mathcal{M}^*(w^*)$ was contained in the model structure, $\mathcal{M}(w)$.

Condition 2: *(Ring networks, Fixed number of data points, fixed noise levels)*

The simulation is intended to investigate whether our approaches still gain advantages when dealing with extremely sparse networks.

100 random networks were generated, from a fixed topology: a ring network. There were 10 nodes in each network and models were generated following the same protocol in condition 1. There was only one input applied to node 1. 65 data points for each node and input were collected for identification and SNR was set to $20dB$.

The result of condition 1 is summarized in Table 6.10.1 and plotted in Figure 6.10.1. Figure 6.10.1 shows that in general, the performance of all the methods degrades as the noise increases except RJMCMC. While Prec of RJMCMC and GESBL stays beyond 90%, Prec of the other methods decays fast as SNR decreases (Prec below 80% at $SNR = -30dB$). When $SNR \geq 10dB$, TPR of all the approaches is robust to the change of the noise level. TPR of RJMCMC is similar with that of GESBL ($TPR > 90\%$) and is higher than the other three methods. When the noise overwhelms the input ($SNR = -30dB$), RJMCMC still retains high TPR (85.9%) whereas TPR of GESBL and the kernel method drops to below 60%. The kernel method is outperformed by RJMCMC as the noise level increases. Nevertheless, it always

provides higher Prec compared with GSBL and SBL. It should be noticed that although TPR of SBL and GSBL is quite high ($> 80\%$) at $SNR = -30dB$, their Prec is very low ($< 65\%$). That means the networks inferred by SBL and GSBL are very dense. It is impossible to tell which links in their inferred networks are correct. It is clear that the performance of RJMCMC is most robust to the noise ($Prec > 90\%,\ TPR > 85\%$). Most true links in the ground truth networks are successfully identified by RJMCMC ($TPR > 85\%$) and the confidence of the inferred networks is very high ($Prec > 90\%$).

Table 6.10.1: Inference of randomly generated ARX networks

	SNR									
	60dB		30dB		20dB		10dB		-30dB	
	Prec	TPR	Prec	TPR	Prec	TPR	Prec	TPR	Prec	TPR
RJMCMC	91.3	94.0	90.4	93.1	90.4	94.0	92.0	93.0	98.1	85.9
Kernel	100	97.3	95.0	92.0	92.9	90.8	88.4	90.8	80.1	53.0
GESBL	100	99.2	100	98.2	100	97.3	100	93.0	99.6	44.5
SBL	100	97.8	99.6	97.2	88.1	95.9	59.5	95.9	53.3	94.9
GSBL	100	76.4	90.8	80.0	81.3	82.2	76.7	82.2	64.4	86.9

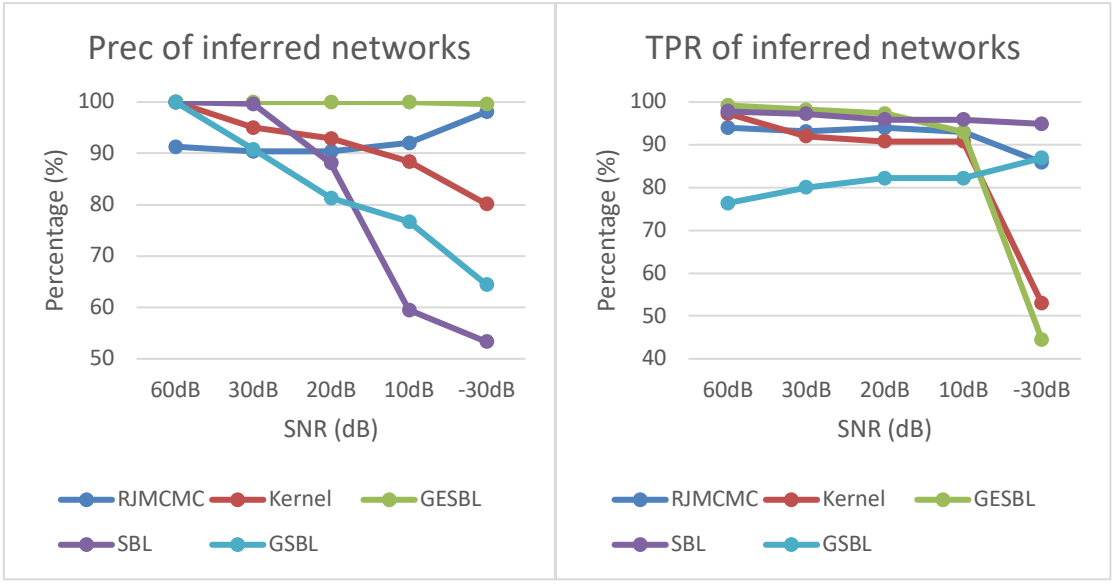


Figure 6.10.1: Precision (*left column*) and True Positive Rate (*right column*) of the inferred ARX networks. The figure displays the result of inferring random ARX networks under different noise levels using various methods. The *x*-axis represents SNR and the *y*-axis denotes Prec or TPR. Inference results of different methods are labelled with different colors.

The inference of ring networks is shown in Table 6.10.2. The ring networks only contain 10 links (of a total of 90 possible links). Hence, high TPR is meaningless unless Prec is also high. Otherwise, it is impossible to tell which inferred links are correct.

The result in Table 6.10.2 highlights the advantage of GESBL over SBL and GSBL. All three methods attain high TPR ($> 90\%$). It should be noticed that GESBL has by far the largest Prec (93.2%), meaning that on average 9.2 links are correctly inferred (out of 10) in a total of

9.9 links estimated (out of 90). Prec of SBL and GSBL is extremely low ($< 20\%$), meaning that these methods estimate almost all 90 possible links. With no doubt, GESBL is superior to SBL and GSBL in inferring extremely sparse networks. Prec of the kernel method (44.6%) is much lower than GESBL (93.2%) although it is higher than SBL (11.2%) and GSBL (13.2%). That implies the kernel method generates a much denser network than a ring structure. It is impossible to tell which inferred links in the network are correct. RJMCMC is no better than GESBL but is superior to the other methods.

Table 6.10.2: Inference of randomly generated ring networks ($SNR = 20dB$)

	$\mathbf{SNR = 20dB}$	
	Prec	**TPR**
RJMCMC	79.4	93.1
Kernel	46.4	95.2
GESBL	93.2	92.4
SBL	11.2	100
GSBL	13.2	97.9

Simulations show that for ARX models, GESBL outperforms all the other methods, especially SBL and GSBL. The performance of RJMCMC and the kernel method degrades as networks become extremely sparse. It is mainly because of the kernel function adopted. The impulse responses of the predictor of ARX models are exactly the polynomial coefficients so the predictor is a FIR system. These finite impulse responses are intrinsically stable. Correlations among different time points of FIRs are negligible. As a result, the stable spline kernel overly constrains impulses responses of ARX models. In contrast, the diagonal covariance matrix applied by GESBL is more appropriate, which treats model parameters as independent random variables. Additionally, since the covariance matrix constructed by the stable spline kernel is only controlled by two hyperparameters, the kernel method and RJMCMC provide a lower degree of freedom for parameter estimation compared with GESBL.

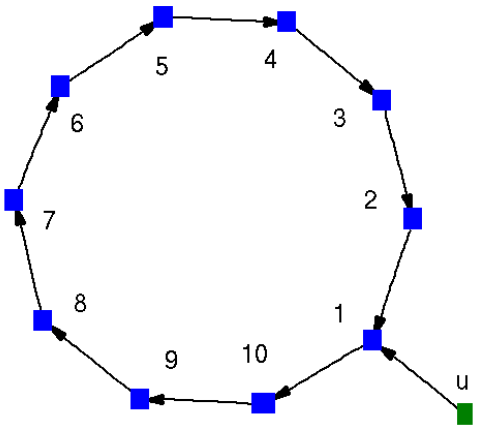


Figure 6.10.1: A ring network. The network contains 10 nodes which form a ring structure. Only one node is driven by an input.

6.10.2 Identify DSF models

In this part of simulations, we identified DSF models. Since the stable infinite impulse responses (IIF) of DSF models were estimated, GESBL, SBL and GSBL were not suitable. Therefore, we concentrated on RJMCMC and the kernel method.

Monte Carlo simulations were conducted to investigate the performance of these two methods under four different conditions.

Condition 1: *(Random sparse networks, Different number of data points, Pure noise)*

This simulation aims to investigate the worst-case scenario where no inputs are driving the system and show how rich data benefit the algorithm performance.

100 networks were simulated whose topology and internal dynamics were generated randomly. All networks contained 10 measured nodes. Each node was independently driven by Gaussian noise with variance 1 (assumed unknown during inference). All DSFs were produced from state space models. To be specific, a sparse stable $A \in \mathbb{R}^{15\times 15}$ matrix was first yielded randomly using function $sprandn(n, n, sparsity)$ in Matlab. The brute force strategy was applied to guarantee that A matrix was Hurwitz (i.e. no eigenvalue was outside the unit circle of the complex plane) and no isolated nodes existed in the network. The structure of one of the generated networks is displayed in Figure 6.10.2. We measured 10 states so that there were 10 manifest nodes in each network and 5 out of 15 were hidden nodes. The length of truncated impulse responses was set to 20. Collected data with different number of points were used for inference.

Condition 2: *(Ring networks, Different number of data points, Pure noise)*

This simulation investigates extremely sparse networks containing feedback loops, which are more challenging to infer compared with condition 1.

The topology of networks was fixed to be a ring as shown in Figure 6.10.1 whereas their internal dynamics were generated randomly. The procedure to generate state space models for these ring networks was the same with condition 1 except that the topology was fixed.

Condition 3: *(Random sparse networks, Different number of data points, Different noise levels)*

This part of simulation aims to investigate how the measured inputs help the inference.

The models generated in condition 1 were employed here. Each measured node was driven by an independent input (known) and noise (unknown). Both inputs and noise were set to be i.i.d. Gaussian with 0 mean. The variance of inputs was 1 while that of the noise varied.

Condition 4: *(Random unidentifiable networks, Different number of data points, No noise)*

In the previous simulations, all the candidate DSF models can be reconstructed from the input-output map (H is diagonal). Now, we consider the situation where DSFs are not unique corresponding to a given transfer matrix G in (6.4.2). We would like to see that with unidentifiable DSF models and under the best-case scenario (no process noise), whether our approaches can still find the ground truth topology.

Models were generated with only one input and without process noise, following the same procedure in condition 1. Therefore, matrix P was a vector which did not meet the conditions in Theorem 5.7.1.

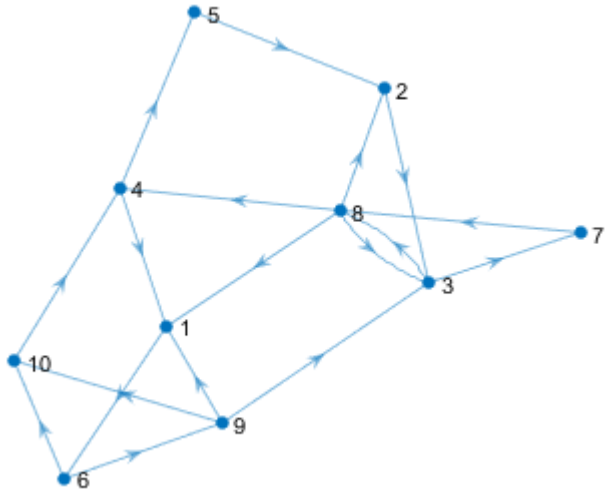


Figure 6.10.2: An example of the randomly generated DSF networks

The result of condition 1 is summarized in Table 6.10.3 and plotted in Figure 6.10.3. It is worth realizing that although DSF models were generated randomly, there were no isolated nodes. Moreover, feedback loops existed in each network. Since networks were driven by pure noise, RJMCMC demands sufficiently long time series data to explore the statistical characteristics of the noise. Prec and TPR of RJMCMC and the kernel method increase as more data points are available for inference. Prec of RJMCMC stays high regardless of the length of data ($> 95\%$). TPR of RJMCMC is low (41.1%) given short time series (100). As more data are used, TPR of RJMCMC increases from 41.1% to 83.9%. Clearly, Prec of RJMCMC is much higher than the kernel method in all cases, indicating RJMCMC is more robust to process noise. Although the highest TPR of RJMCMC (83.9%) is below 90%, it should be noticed that inference is conducted under the worst-case scenario (without inputs). It is expected that with the measured inputs, the performance of RJMCMC can be further improved.

Table 6.10.3: Inference of randomly generated DSF networks

	Number of data points					
	100		**500**		**1000**	
	Prec	TPR	Prec	TPR	Prec	TPR
RJMCMC	97.0	41.1	96.2	73.9	99.1	83.9
Kernel	37.6	70.2	81.7	76.1	88.0	77.1

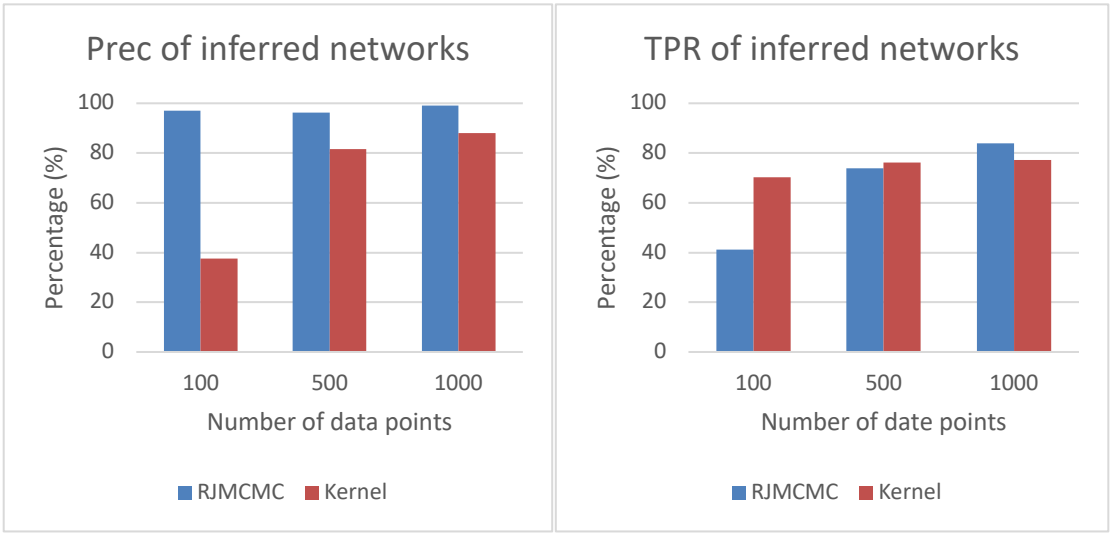


Figure 6.10.3: Precision (*Left column*) and True Positive Rate (*Right column*) of the inferred DSF networks with no input. The figure displays the result of inferring random DSF networks using different length of data. The *x*-axis denotes the number of data points and the *y*-axis presents the value of Prec or TPR. Results of RJMCMC and the kernel method are labelled by different colors.

The result of condition 2 is recorded in Table 6.10.4 and plotted in Figure 6.10.4. As the topology of ring networks is extremely sparse, obtaining high Prec is challenging. The advantage of RJMCMC over the kernel method is obvious. Prec of RJMCMC is always above 95% whereas Prec of the kernel method is below 90%. In addition, TPR of RJMCMC is always higher than the kernel method given sufficient data. Since the noise is unknown, both methods require long time series data to achieve good inference results. With 1000 data points, RJMCMC achieves 96.4% Prec and 89.4% TPR. Although the networks are every sparse, RJMCMC always guarantees high Prec indicating the inferred networks are reliable. The global search of the parameter subspaces encouraged by RJMCMC greatly benefits the detection of network topology. Whereas RJMCMC requires long data sequences to achieve satisfactory performance, data are usually limited in practice. In the next simulation, we will see how the measured inputs help reduce the size of datasets required to achieve good inference results.

Table 6.10.4: Inference of randomly generated ring DSF networks

	Number of data points					
	100		**500**		**1000**	
	Prec	TPR	Prec	TPR	Prec	TPR
RJMCMC	98.1	52.5	98.6	85.6	96.4	89.4
Kernel	24.5	78.5	72.2	85.0	87.2	85.0

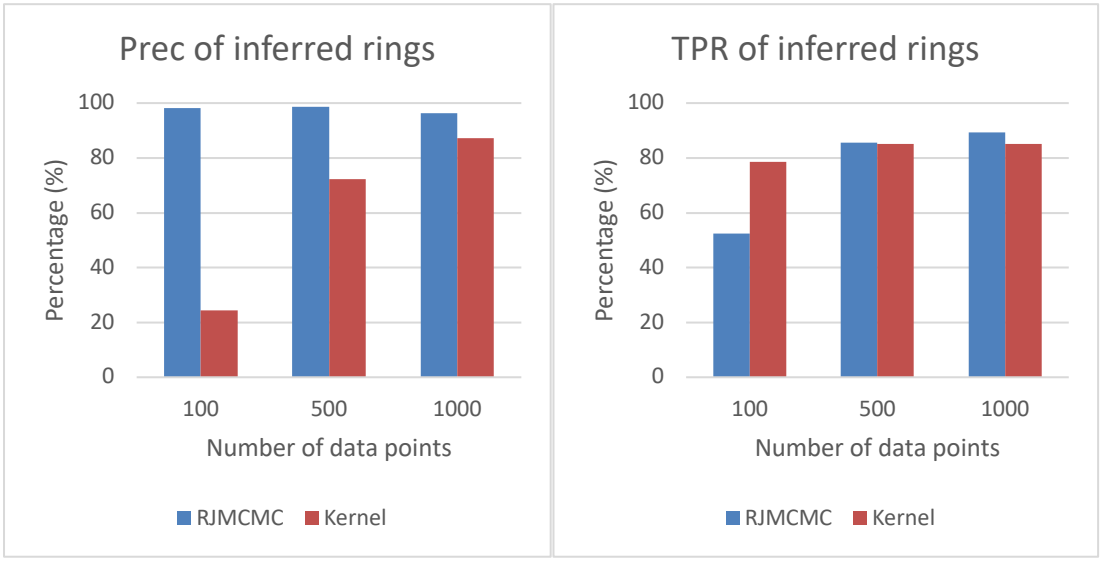


Figure 6.10.4: Precision (*Left column*) and True Positive Rate (*Right column*) of the inferred DSF ring networks with no input. The figure displays the result of inferring ring DSF networks using different length of data. The *x*-axis denotes the number of data points and the *y*-axis presents the value of Prec or TPR. Results of RJMCMC and the kernel method are labelled by different colors.

The result of condition 3 is recorded in Tables 6.10.5-6.10.8 and plotted in Figure 6.10.5. Generally, as SNR decreases (i.e. noise grows), RJMCMC demands more data points to achieve the best performance. It is remarkable that when there is no noise, RJMCMC is able to attain perfect inference (100% Prec and 100% TPR) with only 100 data points. The demand of data is 10 times less than the pure noise case (Table 6.10.4). Hence, inputs tremendously help inference methods counteract the interference caused by noise.

Not surprisingly, as the noise grows, the best performance of RJMCMC degrades. The highest TPR decreases from 100% to 82.3% as SNR decreases to $0dB$. However, the highest Prec for each SNR level is always above 95%. In addition, the shortest length of data necessary to achieve the best performance increases as SNR decreases. RJMCMC requires at least 100, 200, 300 and 600 data points in order to attain good inference results (both Prec and TPR higher than 80%) under different SNR levels, respectively. As the noise increases, the performance of RJMCMC approaches to the worst-case scenario because the contribution of inputs to internal dynamics is overwhelmed by process noise.

Finally, RJMCMC is clearly superior to the kernel method. Although the kernel method provides competitive TPR compared with RJMCMC, RJMCMC presents much higher Prec in all cases, meaning the inferred networks are more reliable. As a result, RJMCMC is more robust to process noise than the kernel method.

Table 6.10.5: Inference of randomly generated DSF networks (No noise)

	Number of data points					
	45		**65**		**100**	
	Prec	TPR	Prec	TPR	Prec	TPR
RJMCMC	99.5	84.0	99.2	97.5	100	100

Kernel	84.7	58.3	91.4	91.7	100	100

Table 6.10.6: Inference of randomly generated DSF networks (SNR=20dB)

	Number of data points					
	65		**100**		**200**	
	Prec	TPR	Prec	TPR	Prec	TPR
RJMCMC	94.0	86.8	95.7	94.2	98.7	96.9
Kernel	70.7	86.7	92.2	91.6	97.4	96.5

Table 6.10.7: Inference of randomly generated DSF networks (SNR=10dB)

	Number of data points					
	100		**200**		**300**	
	Prec	TPR	Prec	TPR	Prec	TPR
RJMCMC	92.0	79.0	93.9	86.1	95.1	88.3
Kernel	74.1	82.5	79.7	88.3	86.3	91.0

Table 6.10.8: Inference of randomly generated DSF networks (SNR=0dB)

	Number of data points					
	400		**500**		**600**	
	Prec	TPR	Prec	TPR	Prec	TPR
RJMCMC	97.3	79.3	97.8	79.9	97.1	82.4
Kernel	78.9	79.1	80.6	78.7	81.7	82.7

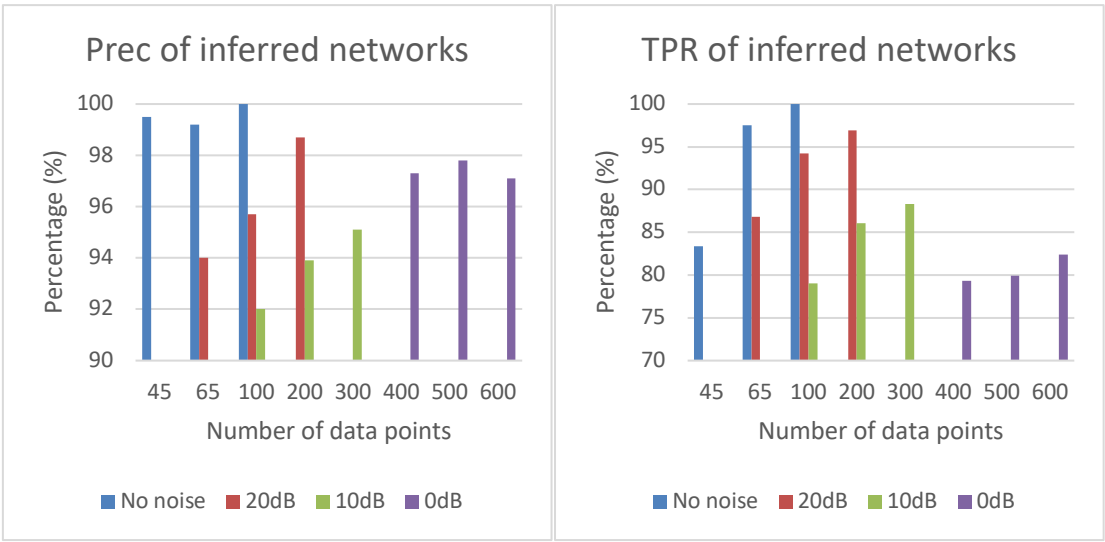


Figure 6.10.5: Precision (*Left column*) and True Positive Rate (*Right column*) of the inferred DSF networks with inputs using RJMCMC. The figure displays the result of inferring randomly generated DSF networks using different length of data under various noise levels. The *x*-axis denotes the number of data points and the *y*-axis presents the value of Prec or TPR. Results under different noise levels are labelled by different colors.

The result of condition 4 is recorded in Tables 6.10.9 and plotted in Figure 6.10.6. The networks were driven by only one input. Hence, DSF models were not unique given the input-

output map. Although prior distributions were imposed to favor sparse topologies, there was no guarantee to capture the ground truth. Prec of RJMMC is always above 80% but TPR is quite low ($< 60\%$). That means RJMCMC tends to produce a sparse network out of all the feasible candidates. However, many true links are missed. Compared with RJMCMC, the kernel method presents the opposite result. The kernel method captures most true links in the network but with a low Prec value. In this case, the inferred networks cannot be trusted. Therefore, RJMCMC is favored over the kernel method since Prec is the first priority in practice.

Table 6.10.9: Inference of unidentifiable DSF networks (No noise)

	Number of data points					
	45		65		100	
	Prec	TPR	Prec	TPR	Prec	TPR
RJMCMC	82.1	46.0	82.9	54.8	80.4	58.1
Kernel	52.3	78.8	57.4	74.4	65.8	74.1

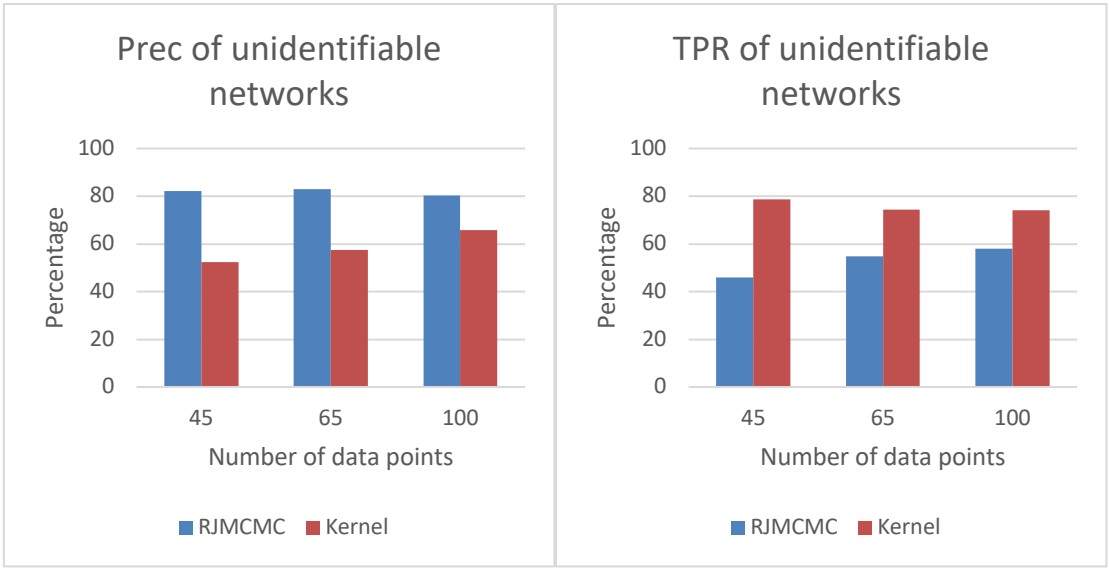


Figure 6.10.6: Precision (*Left column*) and True Positive Rate (*Right column*) of the inferred unidentifiable DSF networks. The figure displays the result of inferring randomly generated DSF networks using different length of data under various noise levels. The *x*-axis denotes the number of data points and the *y*-axis presents the value of Prec or TPR. Results under different noise levels are labelled by different colors.

To conclude, simulations of DSF models indicate RJMCMC is superior to the kernel method. For identifiable DSFs, most true links of the ground truth are identified and the inferred networks are reliable. Even with limited data points, RJMCMC is still able to guarantee high Prec. More importantly, RJMCMC is very robust to process noise. The remarkable performance of RJMCMC is due to its ability to effectively explore the full Bayesian model and encourage a global search of the entire parameter space. The effect of ARD is maximally activated in this case, producing sparse networks. Even if a DSF is unidentifiable, RJMCMC still ensures high confidence of the inferred networks (Prec$> 80\%$) although many true links are missed (TPR$< 60\%$).

6.11 Conclusion

DSF models are used to describe sparse networks, where only a small number of nodes are measurable. A full Bayesian model is established to describe the stochastic property of the target network, where network topology is also assumed to be a random quantity. Instead of approximating the Bayesian model analytically using deterministic approximations, a numerical sampling method called RJMCMC is adopted. Inference of network topology and internal dynamics is based on the empirical distributions built using samples, which accurately reflects the true distributions. The full Bayesian model is explored through a randomized global search of the entire parameter space promoted by RJMCMC. RJMCMC traverses parameter subspaces of different dimensionality to evaluate the probability of different topologies. Simulations show that GESBL outperforms SBL, GSBL, the kernel method and RJMCMC when identifying ARX models whereas RJMCMC is superior to the kernel method, dealing with DSFs. RJMCMC is able to identify most true links in the network and offer reliable inference results.

The main advantage of RJMCMC is that it effectively avoids local maxima of the Bayesian model. Since the true Bayesian model is explored stochastically, RJMCMC allows evaluation of the confidence of inferred networks. In addition, by estimating noise variance through sampling, RJMCMC is robust to process noise. The main drawback of RJMCMC is the high computational cost when inferring large-scale networks.

The future research consists of two aspects. One is to develop a more efficient way to further reduce the computational cost of calculating the inversion and determinant of high-dimensional covariance matrices. The second is to extend RJMCMC to identify nonlinear models. The kernel function has been shown very powerful in characterizing nonlinear functions, which can be used to construct complex nonlinear systems. Therefore, it is believed that the framework of this chapter (i.e. combined RJMCMC and the kernel method) can further benefit the inference of nonlinear networks. However, how to deal with the hidden states of a nonlinear model is still an open question since they cannot be removed as they can in linear models.

Chapter 7.

Network inference of synthesized circadian models

Previous chapters have discussed four different methods of inferring sparse networks: one-to-one, combined group and element sparse Bayesian learning (GESBL), the kernel method and reversible jump Markov chain Monte Carlo (RJMCMC). They are developed sequentially to identify mathematical models of sparse networks with increasing complexity.

Previous simulations show that GESBL, the kernel method and RJMCMC are able to infer networks that accurately fall in their proposed model classes. However, many practical biological networks are highly complex and may not be well approximated by the proposed model. Hence, the key questions are whether these methods can provide reliable inference of real biological networks and what factors can influence their performance.

We take the circadian clock of *Arabidopsis* as a case study. This chapter considers data simulated from existing mathematical models. This provides a controlled environment, where sampling time and noise variance can be adjusted to explore trade-offs between the different methods. In addition, in simulated data, the ground truth is known. In contrast, the next chapter considers real data also from the circadian clock of *Arabidopsis*. Although comparisons between methods are harder, since the true system is unknown, we can still test how different methods perform with real biological data.

To date, around 30 genes have been identified as members of the central oscillator of the circadian clock of *Arabidopsis* [223]. The clock consists of a variety of interlocking feedback loops among the clock genes. However, the exact internal working mechanisms of the clock have not yet been fully understood. A detailed background review of the circadian clock is provided in the next chapter. Here, we mainly introduce the simulation models proposed in literatures.

The model considered in this chapter, which we call the Millar 10 model [61], [62], was generated by fitting nonlinear models to experimental data. A parametrized grey-box nonlinear model was first postulated. The model structure was pre-fixed in accordance to the hypothetical topology based on experimental observations. The model parameters were estimated from data to minimize a designed cost function. The model successfully reflects partial dynamics of the real system, and hence is a good candidate to verify our proposed inference methods. This model has been widely accepted as the platform to test different

inference methods [21], [76].

We use the simulated data of this model for inference. Since the ground truth topology of the network is implied by the model structure, we are able to evaluate the performance of different methods via TPR, FPR and Prec applied in the previous chapters. From Monte Carlo simulations, we can evaluate whether, by applying our developed methods, the inferred circadian clock from real experimental data will be reliable or not.

Section 7.1 introduces the synthesized circadian model: Millar 10 model. Section 7.2 explains the simulation setup. Section 7.3 presents the inference of Millar 10 model using the four proposed methods. Section 7.4 compares our methods with other state-of-the-art methods. Section 7.5 concludes the whole chapter.

7.1 Synthesized models of circadian systems

The circadian system of *Arabidopsis* has 24h rhythms that are entrained by daylight cycles. The circadian system includes complex dynamics among light inputs and interlocking feedback loops of the clock genes. To study the internal dynamics of the circadian system, mathematical models are identified based on experimental data. The established models can be used to predict future behaviors of the circadian system under various conditions. Although the network topology implied by these models may differ from the ground truth, it is expected that they have analogous system dynamics to the circadian system. In particular, the signals are rhythmic, oscillatory and active to light stimuli. Therefore, these models can be treated as artificial circadian systems. It is reasonable to believe that the methods that can accurately infer these synthesized models are also capable of inferring the real circadian system.

One of the synthesized models we will use is Millar 10 [61]. In addition to transcriptional control, this model also simulates post-transcriptional and post-translational regulation within the circadian system. The model contains 7 genes (*LHY, CCA1, PRR7, PRR9, NI, GI, Y* and *TOC1*) and their associated proteins. The system is driven by light signals. The network involves three loops that account for the morning loop, evening loop and interactions between them. In particular, *LHY/CCA1* is regulated by *TOC1* and *PRR7* via a feedforward loop. ZTL is stabilized by *GI* in the presence of light. Dawn and dust sensitivity of the clock is realized by NI-regulated *LHY/CCA1*. The mathematical model is nonlinear and continuous time. The system dynamics are based on Hill functions, Michaelis-Menten kinetics and polynomials that represent transcription, translation and degradation of genes and proteins. Parameters of the model represent the rate of biochemical reactions. Mathematical equations of the model are given below [61]:

(7.1.1)

$$dc_i^m = f_{c_i^m}dt + \sum_{j=1}^{M_{c_i^m}} \frac{f_{c_i^m}(j)}{\sqrt{\left|f_{c_i^m}(j)\right|}} \mathcal{N}_j(0, \sigma dt)$$

$$dc_i = f_{c_i}dt + \sum_{j=1}^{M_{c_i}} \frac{f_{c_i}(j)}{\sqrt{\left|f_{c_i}(j)\right|}} \mathcal{N}_j(0, \sigma dt)$$

where $\mathcal{N}_j(0, \sigma dt)$ are independent Gaussian random variables with zero mean and variance

σdt. c_i^m and c_i represent concentrations of mRNAs and proteins, respectively. $f_{c_i^m}$ and f_{c_i} are functions that describe biochemical reactions. $f(j)$ represents the jth term of function f. Parameters in these functions are biochemical reaction rates whose physical meanings and values can be found in [61]. The SDEs above have similar forms as equation (2.2.6).

Detailed expressions for functions f are listed below. Light signals are denoted by L and D where $L = 1$ represents light and $D = 0$ represents darkness. p, g, m and n are model parameters.

(7.1.2)

$$f_{c_L^m} = \frac{g_1^a}{g_1^a + (c_{P9} + c_{P7} + c_{NI})^a}\left(Lq_1c_P + n_0L + n_1\frac{c_{Tmod}^b}{c_{Tmod}^b + g_2^b}\right) - (m_1L + m_2D)c_L^m$$

$$f_{c_L} = (p_1L + p_2D)c_L^m - m_3c_L - p_3\frac{c_L^c}{c_L^c + g_3^c}$$

$$f_{c_{Lmod}} = p_3\frac{c_L^c}{c_L^c + g_3^c} - m_4c_{Lm}$$

$$f_{c_T^m} = \left(n_2\frac{c_Y^d}{c_Y^d + g_4^d} + n_3\right)\frac{g_5^e}{g_5^e + c_L^e} - m_5c_T^m$$

$$f_{c_T} = p_4c_T^m - (m_6L + m_7D)c_T(c_{ZTL}p_5 + c_{ZG}) - m_8c_T$$

$$f_{c_{Tmod}} = p_{15}\frac{c_T^f}{c_T^f + g_6^f} - (m_{25}L + m_{26}D)c_{Tm}$$

$$f_{c_Y^m} = Lq_2c_P + (n_5L + n_6D)\frac{g_7^s}{g_7^s + c_T^s}\frac{g_{16}^g}{g_{16}^g + c_L^g} - m_9c_Y^m$$

$$f_{c_Y} = p_6c_Y^m - m_{10}c_Y$$

$$f_{c_P} = p_7D(1 - c_P) - m_{11}c_PL$$

$$f_{c_{P9}^m} = Lq_3c_P + n_7\frac{g_8^h}{g_8^h + c_T^h}\frac{c_L^i}{g_9^i + c_L^i} - m_{12}c_{P9}^m$$

$$f_{c_{P9}} = p_8c_{P9}^m - (m_{13}L + m_{22}D)c_{P9}$$

$$f_{c_{P7}^m} = n_8\frac{c_{Ltot}^j}{g_{10}^j + c_{Ltot}^j} + n_9\frac{c_{P9}^k}{g_{11}^k + c_{P9}^k} - m_{14}c_{P7}^m$$

$$f_{c_{P7}} = p_9c_{P7}^m - (m_{15}L + m_{23}D)c_{P7}$$

$$f_{c_{NI}^m} = n_{10}\frac{c_{Lmod}^l}{g_{12}^l + c_{Lmod}^l} + n_{11}\frac{c_{P7}^m}{g_{13}^m + c_{P7}^m} - m_{16}c_{NI}^m$$

$$f_{c_{NI}} = p_{10}c_{NI}^m - (m_{17}L + m_{24}D)c_{NI}$$

$$f_{c_G^m} = Lq_4c_P + \frac{g_{14}^n}{g_{14}^n + c_T^n}\frac{g_{15}^o}{g_{15}^o + c_L^o}n_{12}L - m_{18}c_G^m$$

$$f_{c_G} = p_{11}c_G^m - p_{12}Lc_{ZTL}c_G + p_{13}c_{ZG}D - m_{19}c_G$$

$$f_{c_{ZTL}} = p_{14} - p_{12}Lc_{ZTL}c_G + p_{13}c_{ZG}D - m_{20}c_{ZTL}$$

$$f_{c_{ZG}} = p_{12}Lc_{ZTL}c_G - p_{13}c_{ZG}D - m_{21}c_{ZG}$$

7.2 General procedure

7.2.1 Simulations of the synthesized models

In practice, one type of experimental data available for network inference is microarray data. The expression level of the clock genes is measured every 4 hours for 2 days under constant light. The quality of a dataset is mainly controlled by two factors: sampling frequency and information richness. We did not constrain the simulations of the synthesized models under the same condition of microarray. Instead, we simulated the models under different light conditions and sampled the data with various frequencies. Inference was conducted using four proposed methods in previous chapters. By Monte Carlo simulations, we aim to find out which inference methods should be used in practice and what experimental conditions most benefit inference.

Millar 10 model was simulated for 7 days with 4 days of light-dark cycles followed by 3 days of constant light, where the time unit of the models was hour. The first 2 days of data were discarded to avoid the transition caused by the initial condition. The input to the model was a binary function representing the light signal where 1 denoted light and 0 darkness. The simulated data were sampled every 1 hour (high sampling frequency) and 4 hours (low sampling frequency), respectively. Different time windows were used to select the data for inference, including -48h to -4h (LDLD cycles), -24h to 20h (LDLL cycles), 0h to 44h (LLLL cycles), 24h to 68h (LLLL cycles) and 48h to 92h (LLLL cycles). To enable a fair comparison, the data sampled with the low frequency were interpolated using cubic spline in Matlab at each time unit so that the number of data contained in each time window was the same with the high sampling frequency.

It should be noticed that time windows under different light conditions have different levels of information richness. Time window -48h to -4h includes dynamics of light-dark transitions. Time window -24h to 20h contains information of light-dark transitions and transitions from LD cycles to LL cycles. Time window 0h to 44h has transitions from LD cycles to LL cycles. The rest of the time windows are under the steady state of constant light where light transitions fade away.

In practice, since measuring proteins is expensive, the data of proteins are normally unavailable. Therefore, states representing proteins in the models were treated as hidden nodes and their data were not used for inference.

7.2.2 Performance evaluation

Different methods were used to infer the synthesized model. Since the ground truth topology was known *a priori*, we calculated True Positive Rate (TPR), False Positive Rate (FPR) and Precision (Prec) to evaluate the algorithm performance. To avoid complex comparisons among different methods, we compress the information of TPR, FPR and Prec into two major criteria: the area under the receiver operating characteristic curve (AUROC) and the area under the precision recall curve (AUPREC). A confidence measure is designed to evaluate the reliability of each link in inferred networks. As the threshold for the confidence measure varies, ROC displays the variation of TPR with respect to FPR while PREC displays Prec versus TPR. As a result, a good inference result is indicated by high AUROC and AUPREC.

For the one-to-one method, the confidence of an inferred link is represented by its corresponding model fitness.

GESBL, the kernel method and RJMCMC are developed under the Bayesian framework. Hence, the confidence of inferred links is naturally measured by their marginal posterior probabilities. However, since GESBL and the kernel method apply empirical Bayes to approximate the true distribution, an accurate estimation of the marginal posterior probabilities is not available. Therefore, we resort to the scheme in [21] to evaluate the confidence of inferred links. The confidence of an inferred link $j \to i$ is estimated as:

(7.2.1)

$$P(j \to i|D) = \frac{\left\|w_{j\to i}\right\|_2}{\sqrt{\sum_j \left\|w_{j\to i}\right\|_2^2}}$$

where $w_{j\to i}$ contains the estimated model parameters representing the regulation from node j to node i. For linear ARX models, $w_{j\to i}$ denotes polynomial coefficients (see equation 4.7.2). For DSF models, $w_{j\to i}$ is equivalent to truncated impulse responses (see equation 5.9.1). D denotes the measured data for inference.

The rationale behind (7.2.1) is that link $j \to i$ is more likely to exist if node j contributes more to the regulation of node i compared with all the other nodes.

RJMCMC explores the true probabilistic model directly via numerical sampling so the marginal posterior probabilities of inferred links can be estimated using empirical distributions. For an inferred link $j \to i$, its marginal posterior probability is:

(7.2.2)

$$\begin{aligned} P(j \to i|D) &= \sum_{\pi} P(j \to i, \pi|D) \\ &\approx \frac{1}{N} \sum_{\pi|j\to i\in\pi} \#_\pi \end{aligned}$$

where N is the total number of samples and $\#_\pi$ is the number of samples of topology π.

7.3 Inference of the Millar 10 model

To begin with, the Millar 10 model was simulated 50 times independently. Figure 7.3.1 displays an example of the simulated data. Scale variable σ of the noise variance in (7.1.1) was set to be 2×10^{-3}. The simulated data were sampled at the high (every 1h) or low (every 4h) frequency. Time windows under different light conditions were used to select the data for inference. For each investigated condition (i.e. inference method, time window and sampling frequency), inference of Millar 10 was repeated 50 times, each time using one of the independent simulations of the model. Algorithm performance was evaluated via AUROC and AUPREC. Their values from repeated trials (50 times) were averaged to reduce the uncertainty of a single trial.

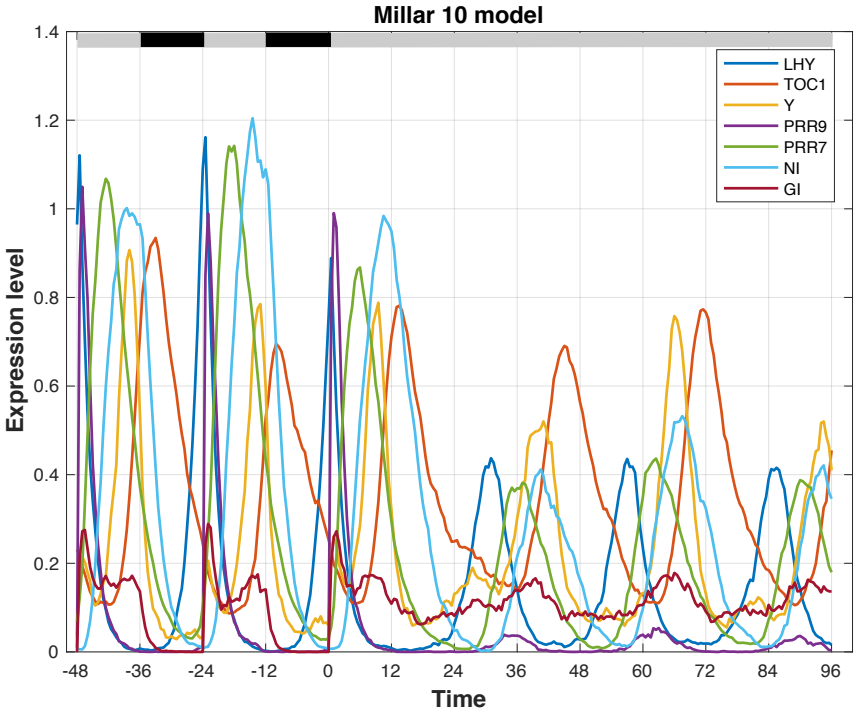


Figure 7.3.1: An example of the simulated data of Millar 10. Colored bars in the graph present light conditions: grey bars for white light and black bars for darkness.

7.3.1 Inference using the one-to-one method

The one-to-one method was used to infer the network. An OE model was identified to describe the interaction between each ordered pair of genes. Model fitness was used as the confidence measure of inferred links.

A heat graph is plotted in Figure 7.3.2 based on model fitness. Each graph shows the result using a specific time window and sampling frequency. Each grid represents a link pointing from a gene labelled in columns to a gene in rows. The brightness of the color is proportional to the value of model fitness. In general, inferred networks using the high sampling frequency are sparser than the low sampling frequency. Time windows under the steady state (constant light) pick more links than time windows that contain light transitions. Some true links in the ground truth are always inferred with high confidence, including (*TOC1, Y*), (*PRR7, PRR9*), (*NI, PRR7*), (*PRR7, LHY*), (*PRR9, LHY*) and (*GI, TOC1*). On the contrary, there are null links in the ground truth that are not correctly identified such as (*TOC1, NI*) and (*NI, GI*), leading to false positives.

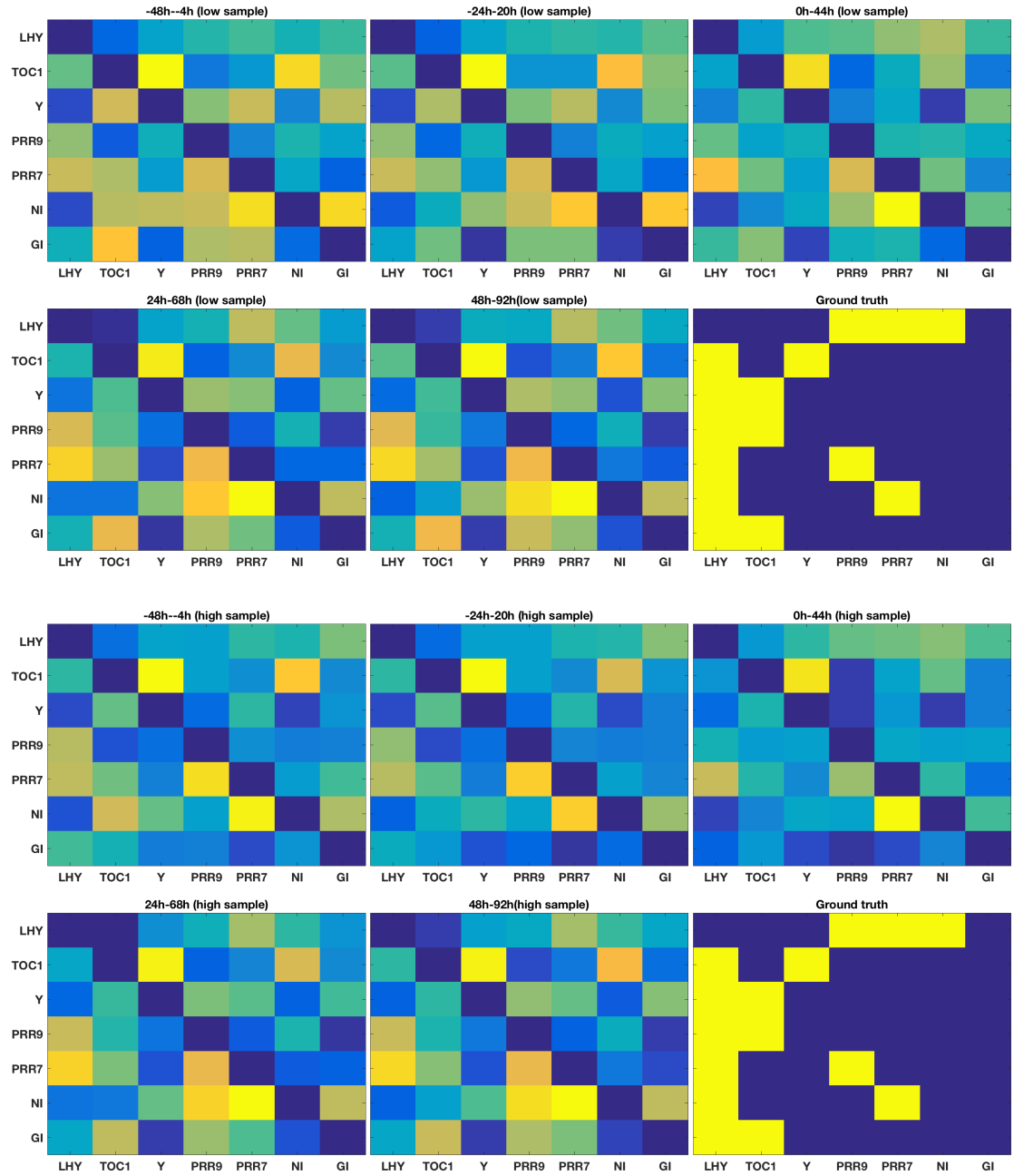


Figure 7.3.2: Heat graph of the inference of Millar 10 using one-to-one method. Each graph shows the result of inference using the data collected with a specified time window and sampling frequency. Each grid presents a link pointing from a gene in columns to a gene in rows. The brightness of the color for each grid is proportional to the model fitness. (X,Y) denotes the link from Y to X.

Figure 7.3.3 displays AUROC and AUPREC of the inferred networks. For the low sampling frequency, time windows under constant light (0h-92h) outperform time windows containing light-dark (LD) cycles. The reason for this is that the Fourier transform of signals in LD contains mostly energy at the frequency associated with 24h. The transient to LL is likely to excite a much wider range of frequencies. In particular, time window 0h-44h, containing the transition from LD to LL, provides the highest AUROC and APPREC ($AUROC = 0.70,\ AUPREC = 0.66$).

Time windows under the steady state are slightly worse than time window 0h-44h. AUPREC of time windows containing LD transitions is below 50%, indicating that the inferred networks are not reliable. At the high sampling frequency, the performance of time window 0h-44h is similar to the low sampling frequency and reserves the best performance. Improvement of time windows under the steady state is very limited. However, AUROC and AUPRED of time windows with LD transitions are greatly improved, which means the one-to-one method takes advantage of the information provided by the additional samples. For high samples, the performance for all time windows is close. However, as seen earlier in chapter 3, one-to-one is not able to fully model complex nonlinear dynamics appearing in some regulations. The main reason is that one-to-one applies a simple model class, i.e. OE models and only adopts low linear model complexity to describe the network.

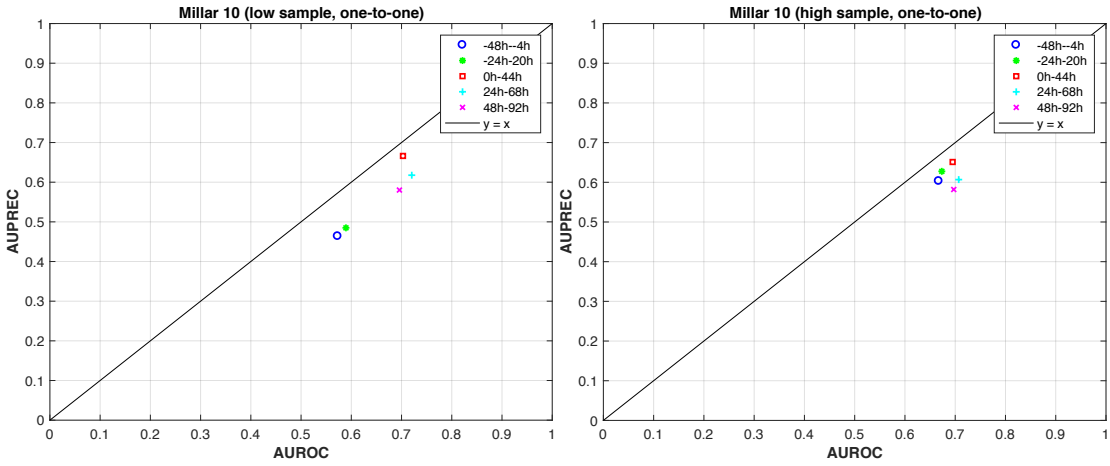


Figure 7.3.3. AUROC versus AUPREC of the inferred Millar 10 model using the one-to-one method. The graph on the left column is the result using the low sampling frequency and the graph on the right is for the high sampling frequency. These graphs display AUROC versus AUPREC with respect to different time windows. Time windows are labelled in the legend. *x*-axis presents AUROC and *y*-axis presents AUPREC.

To conclude, the one-to-one method can effectively infer the Millar 10 model with high AUROC and AUPREC using time window 0h-44h. Many true links in the ground truth are inferred with high confidence and the inferred networks are reliable. Nevertheless, one-to-one cannot fully interpret some complex dynamics of the system due to the limit of 'model power'.

7.3.2 Inference using GESBL

Linear multivariable ARX models were used to describe the network. As the optimal polynomial order for inference is unknown, the polynomial order was treated as a tuning variable during simulations. Low order ARX models cannot approximate complex dynamical systems well whereas high order ARX models can cause over-fitting. To strike a balance between data-fitting and model complexity, the polynomial order was tuned between 10 to 20.

The resulting heat map is plotted in Figure 7.3.4. The confidence of an inferred link is

averaged over different polynomial orders. Compared with the one-to-one method, inferred networks of GESBL are much sparser. In general, inferred networks with the low sampling frequency dataset contain less connectivity than the high sampling frequency. Time windows under the steady state select more links than those involving light transitions. With the low sampling frequency, true links (*PRR9, LHY*), (*PRR7, LHY*), (*NI, PRR7*), (*TOC1, Y)* and (*Y, LHY)* in the ground truth are captured with high confidence depending on the time windows used. However, null link (*Y, PRR7*) is frequently identified as a true link with high confidence. For the high sampling frequency, many regulations from *LHY* are inferred with high confidence regardless of time windows. Inferred networks using time windows with light transitions are extremely sparse. Although time windows under the steady state pick more true links, they also infer null link (*Y, PRR7*) incorrectly, leading to a false positive.

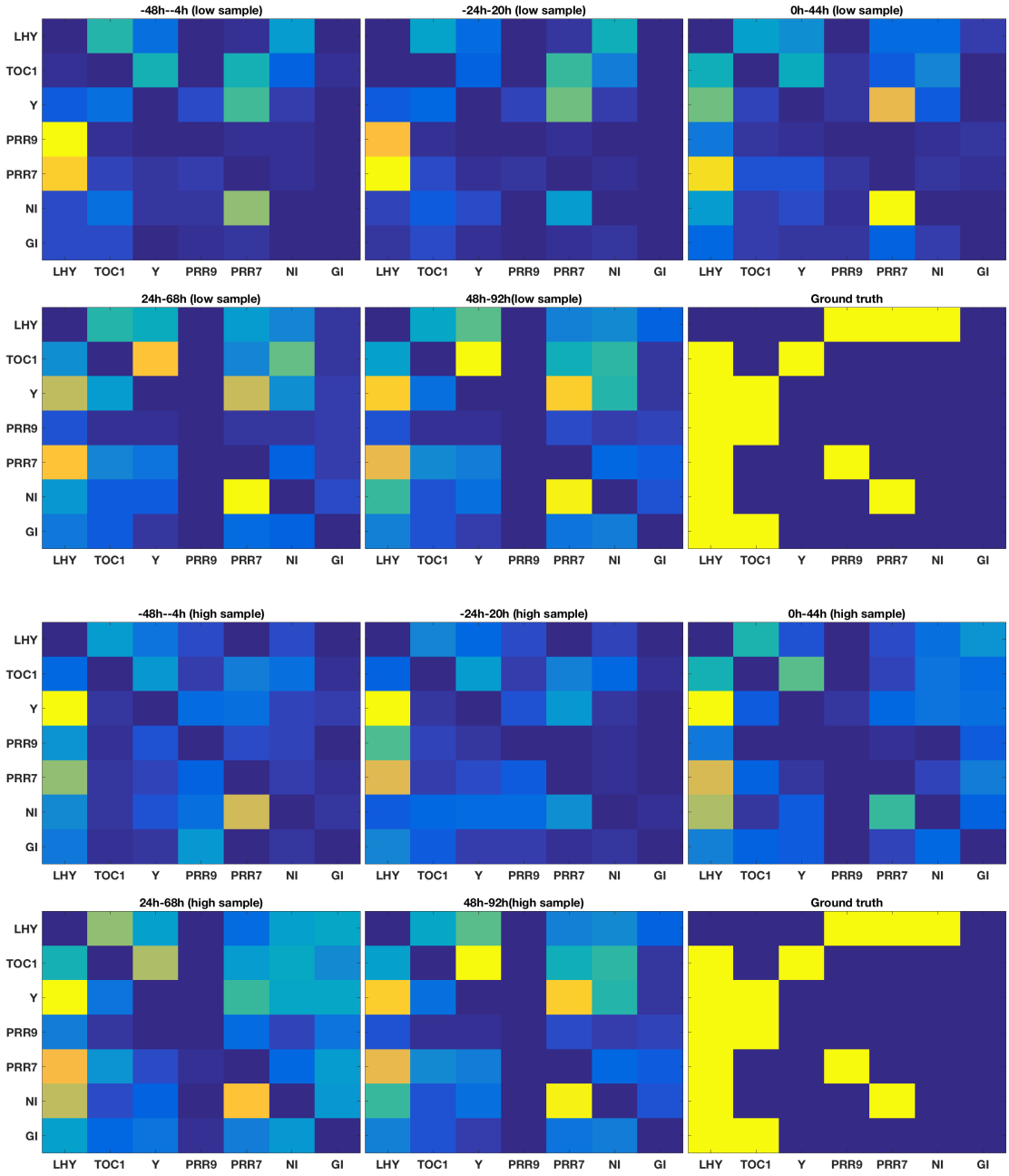


Figure 7.3.4: Heat graph of the inference of Millar 10 using GESBL. Each graph shows the result of inference using the specified time window and sampling frequency. For each graph, grids present links from genes in column to genes in row. The brightness of the color for each grid is proportional to the average confidence of links inferred with different polynomial orders. (X,Y) denotes the link from Y to X.

AUROC and AUPREC resulting from different polynomial orders are plotted in Figure 7.3.5. For each time window, AUROC and AUPREC of all the values of polynomial orders are displayed in the same color. For the low sampling frequency, all points gather closely suggesting that the algorithm performance is relatively robust to polynomial orders. Time windows containing light transitions (0h-44h) are slightly better than those under the steady state. In particular, AUROC and AUPREC of time window 0h-44h are always above 60% and 50%, respectively. For the high sampling frequency, GESBL becomes more sensitive to polynomial orders. Performance of time windows is not significantly improved compared with the low sampling frequency. Time window 0h-44h outperforms the other time windows. In general, the one-to-one method outperforms GESBL equipped with linear ARX models.

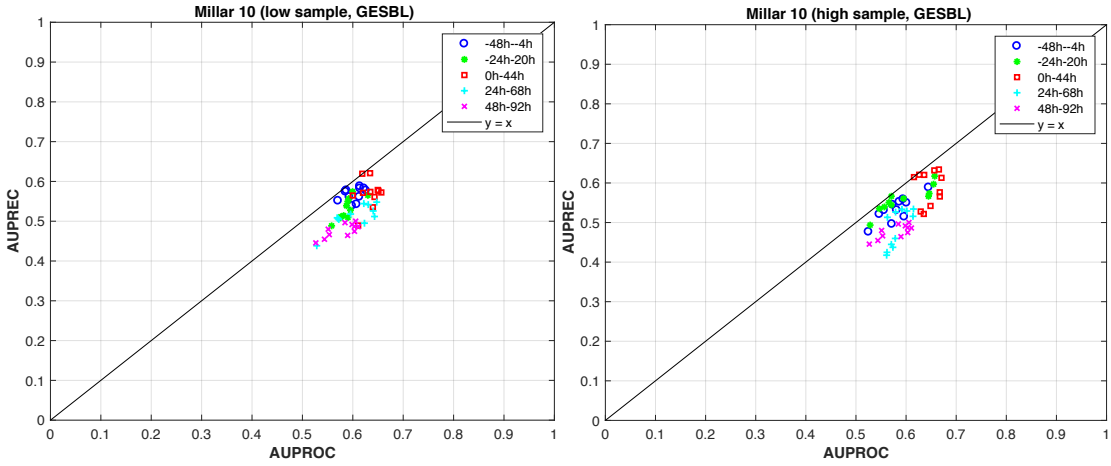


Figure 7.3.5. AUROC versus AUPREC of the inferred Millar 10 model using GESBL. The graph on the left column is the result using the low sampling frequency and the graph on the right is for the high sampling frequency. These graphs display AUROC versus AUPREC with respect to different time windows. Time windows are labelled in the legend. For each time window, AUROC and AUPREC with respect to different polynomial orders are displayed in the same color. *x*-axis presents AUROC and *y*-axis presents AUPREC.

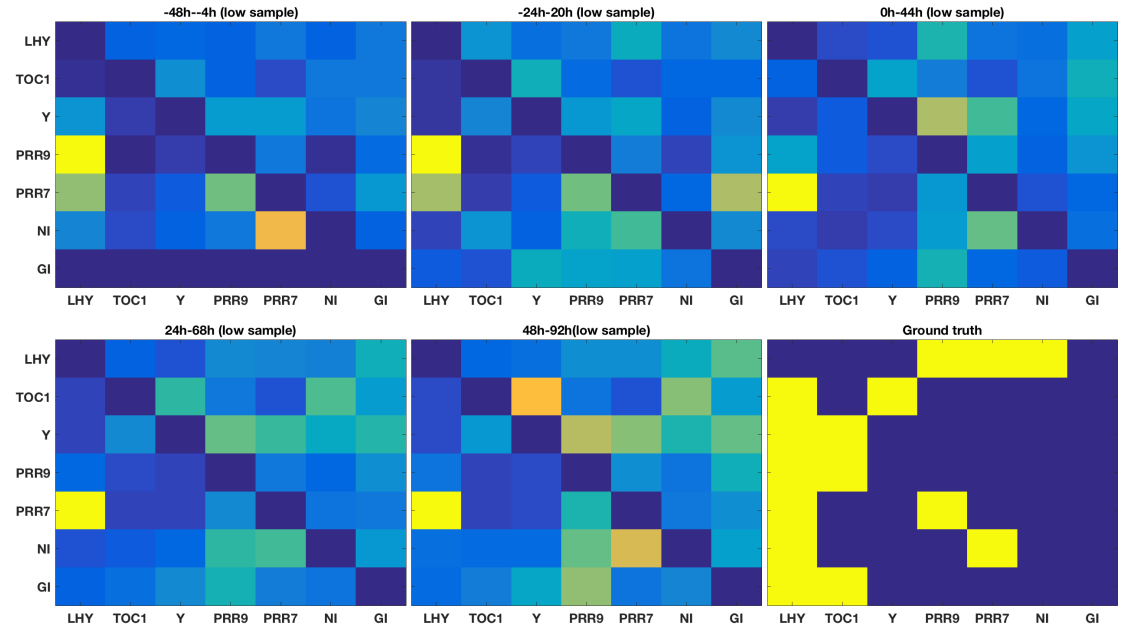
To conclude, GESBL is able to produce sparse networks. However, the method is sensitive to the choice of polynomial orders of linear ARX models. In fact, simulations indicate that GESBL fails to enforce model parsimony. Since a linear ARX model is used to approximate the nonlinear Millar model, the polynomial order of the model is not upper bounded and model parameters are no longer element sparse. However, as the polynomial order increases, over-fitting occurs. The auto-regression term of an ARX model compensates for the contribution from other nodes leading to an extremely sparse topology. Therefore, determination of the polynomial order is problematic in practice. Overall, GESBL equipped with linear ARX models

is not suitable to infer complex biological networks. However, we will see later that, by using a grey-box nonlinear model, the performance of GESBL is tremendously improved.

7.3.3 Inference using the kernel method

The kernel method applied a DSF model to describe the network. The model was expressed in a non-parametric way using impulse responses. For the implementation purpose, impulse responses were truncated. As the optimal truncation length was unknown, truncation length was treated as a tuning variable. To have a fair comparison with GESBL, the length of truncated impulse responses was also tuned between 10 and 20.

A heat map is plotted in the same way as GESBL in Figure 7.3.6. The generated networks are very sparse. For the low sampling frequency, the true links in the ground truth including (*PRR9, LHY*), (*PRR7, LHY*), (*TOC1, Y*) and (*NI, PRR7*) are identified with high confidence depending on the time windows used. All null links from *GI* are identified with high confidence. Nevertheless, almost all regulations from *TOC1* are missed. For the high sampling frequency, more true links are identified. In general, time windows under constant light infer more links than those with LD transitions. However, link (*GI, LHY*) is only inferred by the time window under LDLD. Null links (*Y, GI*), (*Y, NI*) and (*PRR9, PRR7*) are frequently inferred incorrectly by time windows under constant light. With the high sampling frequency, time windows have a great impact on inference. That is because information loss of the collected data is small and DSFs are capable of describing complex dynamics.



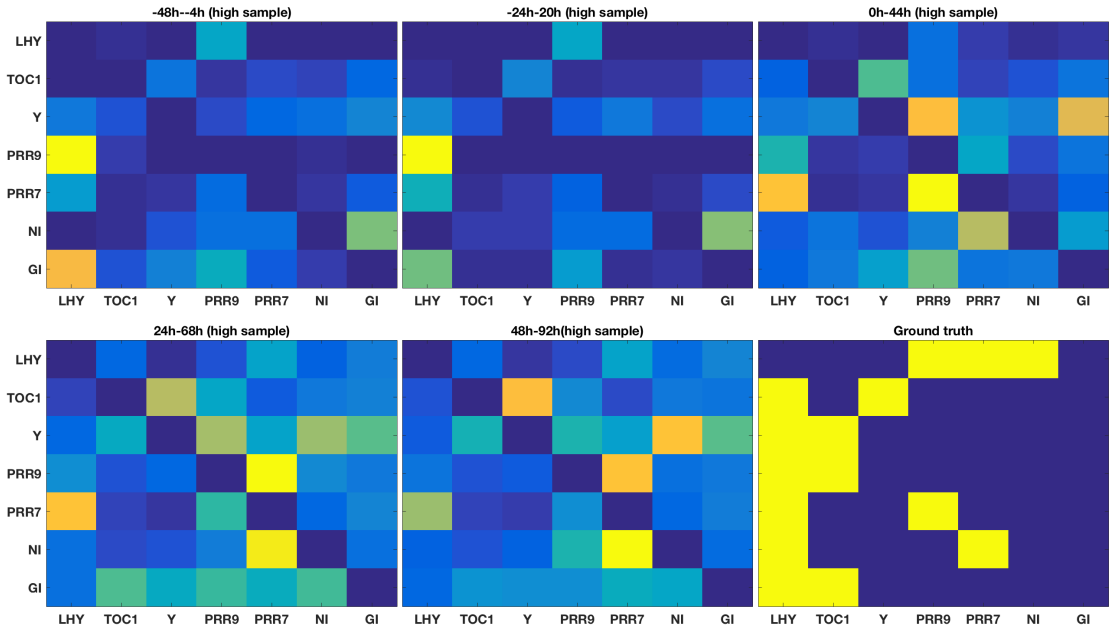


Figure 7.3.6: Heat graph of the inference of Millar 10 using the kernel method. Each graph shows the result of inference using a particular time window and sampling frequency. For each graph, grids represent links pointing from genes in column to genes in row. The brightness of the color is proportional to the average confidence of links inferred with different truncation length of impulse responses. (X,Y) denotes the link from Y to X.

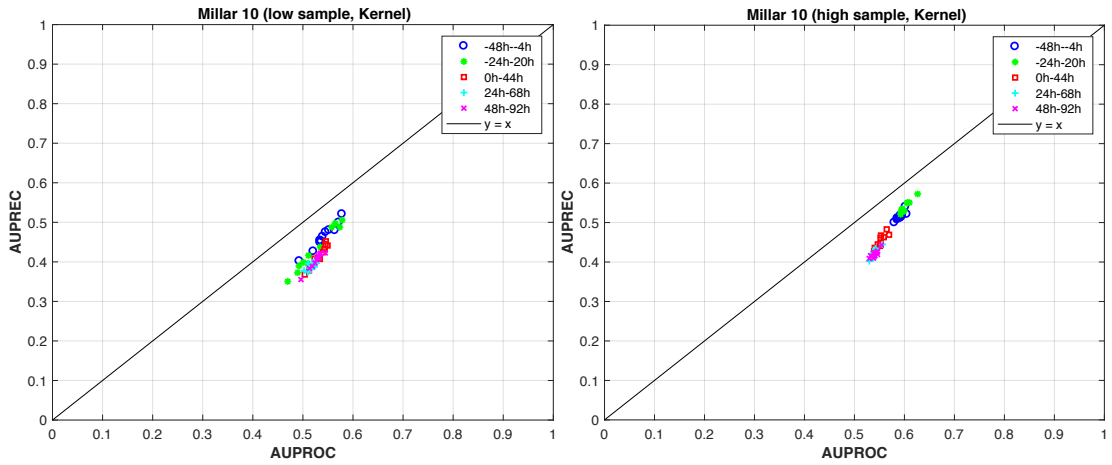


Figure 7.3.7. AUROC versus AUPREC of the inferred Millar 10 model using the kernel method. The graph on the left column is the result using the low sampling frequency and the graph on the right is for the high sampling frequency. These graphs display AUROC versus AUPREC with respect to different time windows. Time windows are labelled in the legend. For each time window, AUROC and AUPREC with respect to different truncation length of impulse responses are displayed in the same color. *x*-axis presents AUROC and *y*-axis presents AUPREC.

AUROC and AUPREC are displayed in Figure 7.3.7. Compared with GESBL, the kernel method is much more robust to the truncation length of impulse responses in all cases because all points of each time window gather closely. Therefore, the problematic tuning can be avoided

in practice since truncation length does not have a great impact on the final result. For the low sampling frequency, time windows under LDLD and LDLL are slightly better than the others. Time windows under constant light have very similar performance. In general, AUROC and AUPREC of all time windows are both below 60%. The high sampling frequency improves the algorithm performance. The kernel method becomes even more robust to the truncation length since all points of each time window gather compactly. Roughly speaking, AUROC and AUPREC of each time window are both increased by around 5% compared to the low sampling frequency. Obviously, time windows under LDLD and LDLL outperform the others. Their AUROC and AUPREC are around 60% and 51%, respectively.

To conclude, the main advantage of the kernel method is that it is robust to the tuning variable (truncation length of impulse responses), which avoids the problematic tuning procedure. In addition, with the high sampling frequency, the kernel method can recover many true links ($\mathrm{AUROC} \approx 60\%$). Time windows under LDLD and LDLL are recommended to achieve the best performance. Nevertheless, the inferred networks are not reliable ($\mathrm{AUPREC} \approx 50\%$). It is difficult to tell which inferred links are correct. The kernel method has to solve a highly nonlinear optimization problem where the sparsity pattern of the solution determines network topology. Hence, the kernel method is very sensitive to local optimal solutions, which influences the accuracy of inferred networks.

7.3.4 Inference using RJMCMC approach

To apply the RJMCMC method, the network was described by a DSF. Impulse responses were assumed to be Gaussian processes with the stable spline kernel as the covariance function. Network topology was treated as a random quantity. Numerical sampling was used to estimate empirical posterior distributions of network topology. For implementation purposes, impulse responses were truncated. As the optimal truncation length was unknown, truncation length was treated as a tuning variable. To be consistent with the kernel method, the length of truncated impulse responses was tuned between 10 to 20.

A heat map of the inference is plotted in the same way as the kernel method in Figure 7.3.8. The inferred networks are very sparse. For the low sampling frequency, true links in the ground truth including (*PRR7, LHY*), (*PRR9, LHY*), (*TOC1, Y*) and (*NI, PRR7*) are successfully inferred with high confidence, which, however, depend on time windows. Null links from *GI* are correctly identified. However, most regulations from *TOC1* are missed. For the high sampling frequency, more true links are identified with high confidence such as (*LHY*, *PRR9*), (*GI, LHY*) and (*PRR7, PRR9*). Nevertheless, null links (*Y, PRR7*) and (*NI, PRR9*) are frequently inferred as true links by some time windows, leading to false positives.

AUROC and AUPREC are displayed in Figure 7.3.9. RJMCMC is very robust to the truncation length of impulse responses as all points of each time window gather closely. Hence, similar to the kernel method, tuning truncation length is not essential in practice since truncation length does not influence the final result critically. The accuracy of inferred networks is greatly improved compared with the kernel method. For the low sampling frequency, the performance of time windows containing light transitions is similar. These time windows outperform the others under the steady state. Time windows under the steady state achieve around 60% AUROC and 52% AUPREC whereas time windows containing light transitions attain about 63% AUROC and 58% AUPREC. For the high sampling frequency, the

performance of all time windows is further improved. AUROC and AUPREC of time windows under constant light are slightly increased. AUROC and AUPREC of time windows under LDLL reach above 65% and 60%, respectively. The time window under LDLL outperforms all the others, achieving 70% AUROC and 69% AUPREC as the best result.

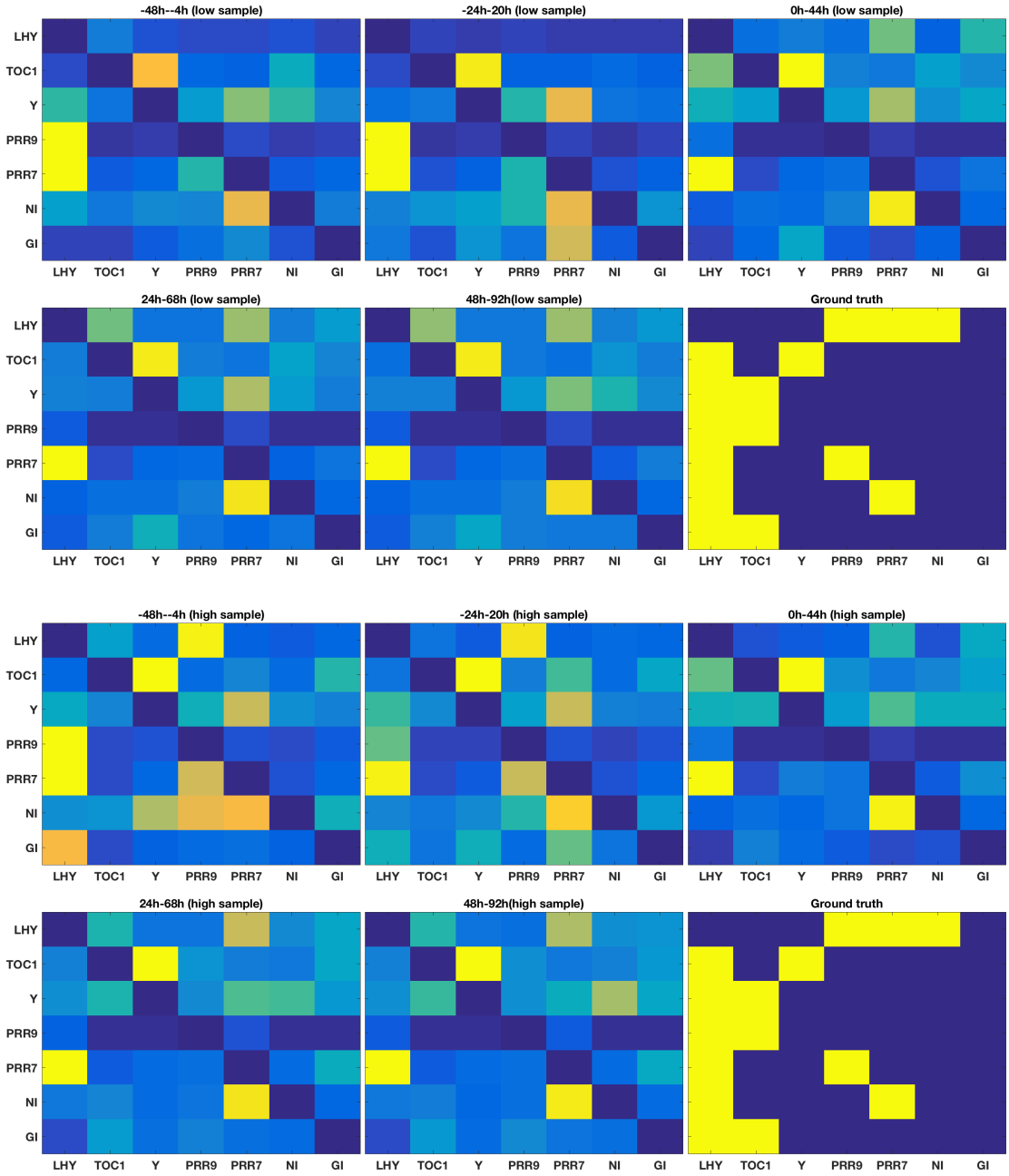


Figure 7.3.8: Heat graph of the inference of Millar 10 using RJMCMC. Each graph shows the result of inference using a specified time window and sampling frequency. For each graph, grids represent links pointing from genes in column to genes in row. The brightness of the color for each grid is proportional to the average confidence of links inferred with different truncation length of impulse responses. (X,Y) denotes the link from Y to X.

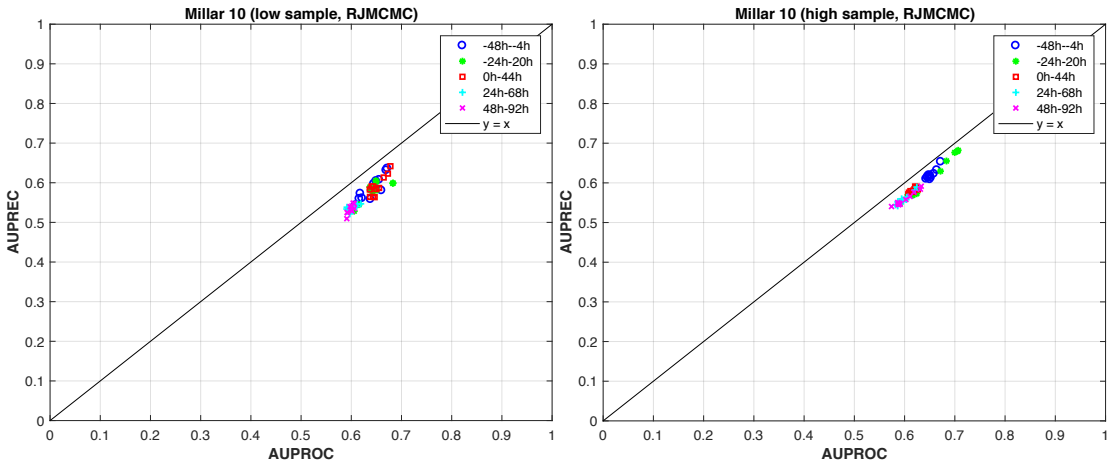


Figure 7.3.9. AUROC versus AUPREC of the inferred Millar 10 model using RJMCMC. The graph on the left column is the result using low sampling frequency and the graph on the right is for the high sampling frequency. These graphs display AUROC versus AUPREC with respect to different time windows. Time windows are labelled in the legend. For each time window, AUROC and AUPREC with respect to different truncation length of impulse responses are displayed in the same color. *x*-axis presents AUROC and *y*-axis presents AUPREC.

To conclude, RJMCMC generates sparse network topologies. The performance of the method is robust to truncation length of impulse response, which avoids tuning in practice. The preference of RJMCMC over the high sampling frequency and the time window under LDLL suggests DSF models are very powerful in capturing system dynamics in response to complex light transitions. The improved performance compared with the kernel method indicates that RJMCMC effectively encourages a global search of the optimal topology, thus reducing the risk of being trapped at local maxima of distributions.

7.4 Comparison with state-of-the-art methods

Inference of the circadian clock of *Arabidopsis* has been intensively studied in the past few years. A variety of inference methods have been developed and some success has been achieved. In particular, the Millar 10 model has been frequently used as a platform to test inference methods. The author of a recently published paper, [76], proposed a novel inference method called iCheMA to infer the circadian clock. The method was tested on the Millar 10 model and compared with a variety of state-of-the-art methods including hierarchical Bayesian regression (HBR), LASSO, elastic net, etc. The author claimed that iCheMA outperformed most existing methods above. Therefore, this chapter compares our methods with iCheMA.

7.4.1 Simulation setup

Millar 10 was simulated with the increased process noise so that the simulated data were comparable with the real microarray data. The noise variance was 5 times higher than that of the last section ($\sigma = 10^{-2}$). Simulations were conducted 50 times independently, generating 50 groups of time series for inference. Figure 7.4.1 displays one example of such simulations. Data were collected with the low (every 4h) and high (every 1h) sampling frequency. To have

a fair comparison, data under the low sampling frequency were interpolated at each time unit (h) so that the number of data points was the same with the high sampling frequency. Due to the high computational cost of iCheMA, we only considered time window 0h-44h (consistent with the real experimental condition) and time window -24-20h (containing the most complex nonlinear dynamics).

Inference of the model was repeated 50 times for all methods, each time using one simulated dataset. The average of AUROC and AUPREC over 50 trials was calculated to evaluate the algorithm performance. Next, we briefly explain iCheMA and compare it to our methods.

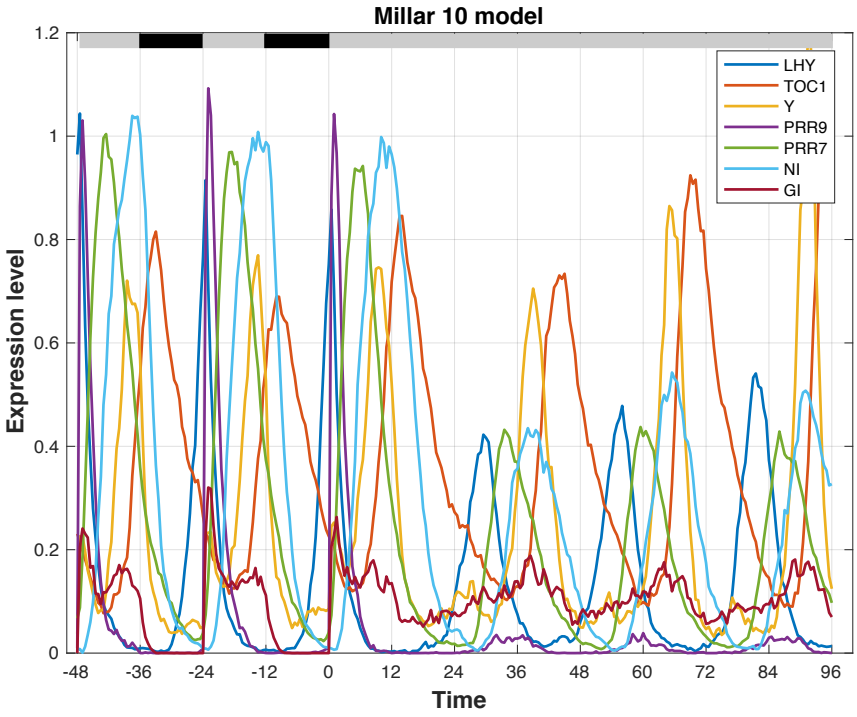


Figure 7.4.1: An example of the simulated data of Millar 10. Colored bars in the graph present light conditions: grey bars for white light and black bars for darkness.

7.4.2 iCheMA

iCheMA postulates a simple grey-box physical model to represent the circadian clock. Simplified Michaelis-Menten kinetics are used to describe biological processes. The model is built on the transcriptional level, where the dynamics of proteins are not considered. The dynamical network involving mRNA degradation and transcriptional regulation is expressed by a series of ODEs. For the ith clock gene, the model is written as [76]:

(7.4.1)

$$\frac{dx_i(t)}{dt} = -v_{0,i}x_i(t) + \sum_{u\in\pi_i} v_{u,i}\frac{I_{u,i}x_u(t) + (1-I_{u,i})k_{u,i}}{x_u(t) + k_{u,i}}$$

where x_i denotes a clock gene. Degradation of mRNAs is described by linear terms and transcriptional regulation is characterized by Michaelis-Menten kinetics. $v_{u,i}$ and $k_{u,i}$ are maximum reaction rate and Michaelis-Menten parameter, respectively. They are estimated during inference. π_i is a set of clock genes that regulate the ith clock gene. Since the ground

truth connectivity is unknown, π_i also needs to be determined during inference. The indicator function, $I_{u,i}$ shows whether the regulation is active ($I_{u,i} = 1$) or repressive ($I_{u,i} = 0$).

The model in (7.4.1) can be rewritten in a regression form. However, the resulting regression model is not linear in parameters. The derivative of $x_i(t)$ is treated as the output of the regression model and estimated using Gaussian process with the radial basis kernel during inference [76].

iCheMA introduces priors and hyperpriors for (7.4.1) to establish a full Bayesian model. iCheMA explores the network topology and internal dynamics of Millar 10 by exhaustively investigating each possible set π_i along with all combinations of indicator functions, $I_{u,i}$. The true posterior distribution of the probabilistic model is evaluated by numerical sampling using the traditional MCMC method. To reduce the high computational cost arising from the combinatorial search, the maximum cardinality of π_i is set to 3 by iCheMA.

The marginal posterior probability for gene j being the regulator of gene i is evaluated via 'model averaging':

(7.4.2)

$$P(j \to i|D) = \frac{\sum_{\pi_i|j\in\pi_i} P(D|\pi_i)P(\pi_i)}{\sum_{\pi_i} P(D|\pi_i)P(\pi_i)}$$

where D denotes the measured data. Calculating (7.4.2) requires to integrate the product of marginal likelihood $p(D|\theta,\pi_i)$ and prior distribution $p(\theta|\pi_i)$ with respect to θ where θ contains model parameters. Nevertheless, only samples from posterior distribution $p(\theta|D,\pi_i)$ are available. Hence, marginal conditional distribution $P(D|\pi_i)$ cannot be evaluated directly. To estimate $P(D|\pi_i)$, Chib's method is applied [224].

7.4.3 GESBL

To have a fair comparison with iCheMA, instead of using linear ARX models, we adopt the same model (7.4.1) to represent the network. Extension of GESBL to nonlinear ARX models has been discussed in chapter 4. While iCheMA implements a combinatorial search of basis functions (i.e. linear functions and Michaelis-Menten kinetics), GESBL selects basis functions in a single run by imposing sparsity to the model parameters. A basis function is assumed to be appropriate if its estimated weight parameter $v_{u,i}$ is non-zero. One advantage of GESBL over iCheMA is that there is no need to set an upper for the cardinality of π_i since all clock genes are considered simultaneously.

It should be noticed that parameter $k_{u,i}$ is not linear with respect to the basis function. Although $k_{u,i}$ can be treated as an unknown deterministic variable and estimated using empirical Bayes, the resulting optimization problem turns out to be highly nonlinear. Consequently, the update of hyperparameters using the EM algorithm does not have a closed form. To simplify the problem, we do not estimate $k_{u,i}$ explicitly during inference. Instead, each basis function is expanded into a group of its counterparts, within which each element is endowed with a particular value of $k_{u,i}$. For the low sampling frequency, $k_{u,i}$ ranges from 0.5 to 2 with an increment of 0.5. For the high sampling frequency, $k_{u,i}$ is expanded as from 0.5 to 5 with an increment of 0.5. The simulations later show that the inference result is robust to the setting of $k_{u,i}$. Consequently, the resulting model can be written as:

(7.4.3)

$$\frac{dx_i(t)}{dt} = -v_i x_i(t) + \sum_{u=1,u\neq i}^{m} \sum_{j=1}^{n} v_{uj}^1 \frac{x_u(t)}{x_u(t)+K_j} + v_{uj}^2 \frac{K_j}{x_u(t)+K_j}$$

where m is the total number of clock genes and K is a finite set of values of $k_{u,i}$. With the notation of Maltab, $K = 0.5{:}\,0.5{:}\,2$ for the low sampling frequency and $K = 0.5{:}\,0.5{:}\,5$ for the high sampling frequency. K_j denotes the jth element of K. Note that basis functions contain Michaelis-Menten kinetics of both active and repressive forms.

The model in (7.4.3) is then recast into a linear regression form as:

(7.4.4)

$$\frac{dx_i(t)}{dt} = \Phi\theta$$

where

$$\begin{aligned}
\Phi &= [\phi_1 \quad \cdots \quad \phi_m] \\
\phi_{u\neq i} &= \left[\frac{x_u}{x_u+K_1} \quad \frac{K_1}{x_u+K_1} \quad \cdots \quad \frac{x_u}{x_u+K_q} \quad \frac{K_q}{x_u+K_q}\right] \\
\phi_i &= -x_i \\
\theta &= [\theta_1 \quad \cdots \quad \theta_m]' \\
\theta_{u\neq i} &= \left[v_{u1}^1 \quad v_{u1}^2 \quad \cdots \quad v_{un}^1 \quad v_{un}^2\right] \\
\theta_i &= v_i
\end{aligned}$$

Note that the derivative in (7.4.4) is estimated using Gaussian process as iCheMA. As GESBL applies empirical Bayes to approximate the true distribution, an accurate estimation of the marginal posterior probability of regulations is not available. Therefore, we resort to the same scheme in the previous sections to evaluate the confidence of inferred links. The marginal posterior probability of link $j \to i$ is estimated as:

(7.4.5)

$$P(j \to i|D) = \frac{\|\theta_j\|_2}{\|\theta\|_2}$$

7.4.4 Other methods

The one-to-one method is used similarly as before. For the kernel and RJMCMC methods, a DSF is used to describe the Millar 10 model. Since previous simulations indicate that truncation length of impulse responses has little impact on the final result, the length of truncated impulse responses is set to 10.

7.4.5 Results

The average AUROC and AUPRC of 50 independent trials using each method with time window 0h-44h and -24h-20h are plotted in Figure 7.4.2 and 7.4.3, respectively. Figure 7.4.2 shows that for the low sampling frequency, the performance of one-to-one and RJMCMC is similar where one-to-one achieves higher AUPREC (64%) and RJMCMC attains higher AUROC (65%). These two methods are obviously superior to the others. The performance of iCheMA is better than the kernel method and GESBL, achieving 60% AUROC and 51% AUPREC. Among all methods, GESBL presents the worst result with 53% AUROC and 45% AUPREC. For the high sampling frequency, the performance of all methods is improved, especially for

GESBL and iCheMA. In this case, GESBL achieves the highest AUROC and AUPREC at 76% and 77%, respectively. The performance of iCheMA is greatly improved with AUROC and AUPREC around 69% and 67%, respectively. The performance of RJMCMC and one-to-one is very close to iCheMA, whose AUROC and AUPREC are both above 65%. The performance of the kernel method is slightly improved but outperformed by the other methods.

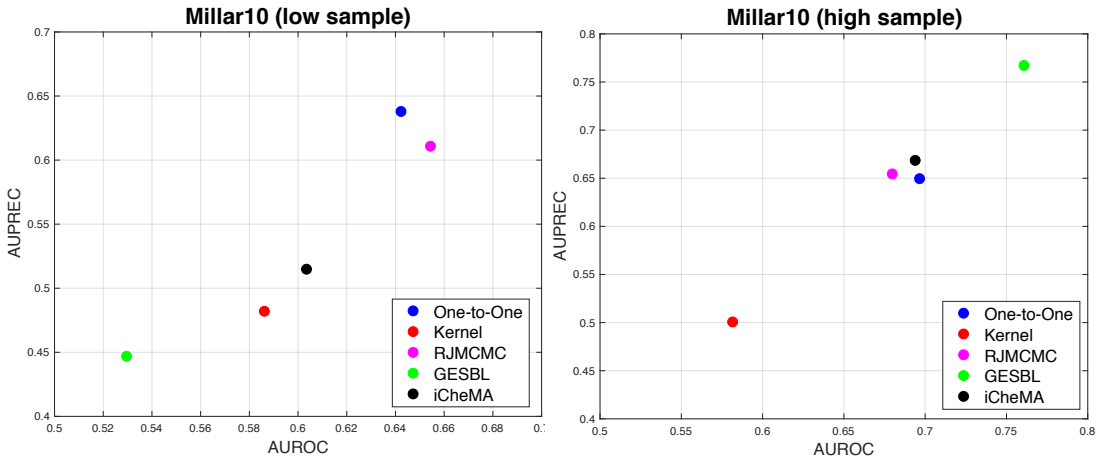


Figure 7.4.2: AUROC and AUPREC of five inference methods using time window 0h-44h. Points in the graph are the averaged AUROC and AUPRC of 50 independent trials.

For time window -24h-20h in Figure 7.4.3, AUROC and AUPREC of all methods are below 60% for the low sampling frequency. RJMCMC provides the best result with 59% AUROC and 52% AUPREC. One-to-one is the top-two method, followed by GESBL. The performance of iCheMA and the kernel method is equally poor, whose AUROC and AUPREC are both below 50%. Compared with time window 0h-44h, the performance of all methods generally degrades. The reason is that information loss of the data at the low sampling frequency disables inference methods to explore complex nonlinear dynamics. For the high sampling frequency, the accuracy of inferred results is critically improved. RJMCMC presents the highest AUROC (75%) and AUPREC (73%). GESBL is the top-two method. iCheMA and one-to-one have similar performance. The kernel method is no better than the other methods.

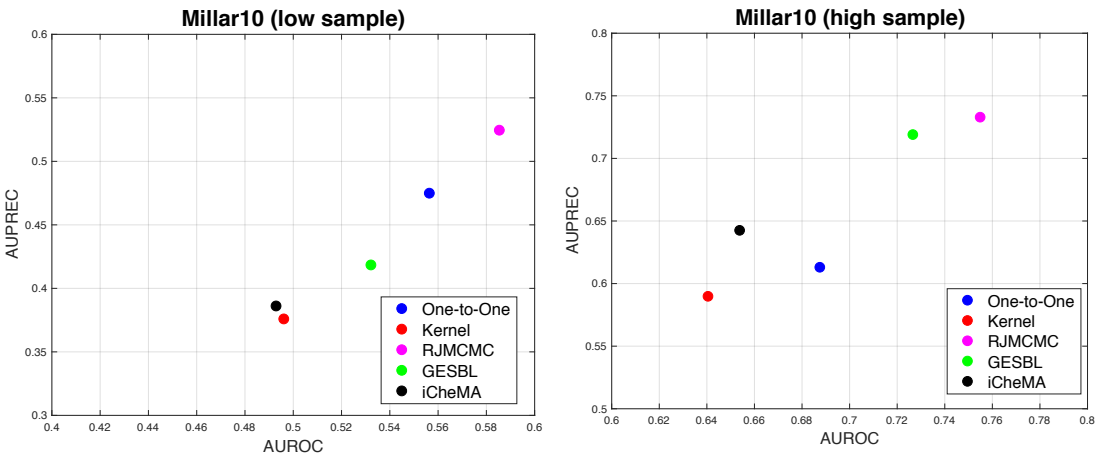


Figure 7.4.3: AUROC and AUPREC of five inference methods using time window -24h-20h. Points in the graph are the averaged AUROC and AUPRC of 50 independent trials.

To evaluate the estimation accuracy of the system dynamics of Millar 10, the identified models were validated on a new dataset that was not used for inference. The new dataset was simulated following the same procedure in this chapter. The models were used to predict the dataset (applying the corresponding predictors discussed in the previous chapters). The predicted outputs are compared with the true ones using the following criterion:

(7.4.6)

$$Fitness = 100\left(1 - \frac{\|y - \hat{y}\|}{\|y - \bar{y}\|}\right)$$

where y denote the collected data (new dataset) of a node, $\hat{y}$ are the predicted outputs and $\bar{y}$ are the mean of the collected data. Fitness of each node is averaged over 50 independent trials.

Since one-to-one builds a model for each ordered pair of nodes independently, the estimated models cannot be used for prediction. Hence, only the kernel method, GESBL, RJMCMC and iCheMA are under consideration.

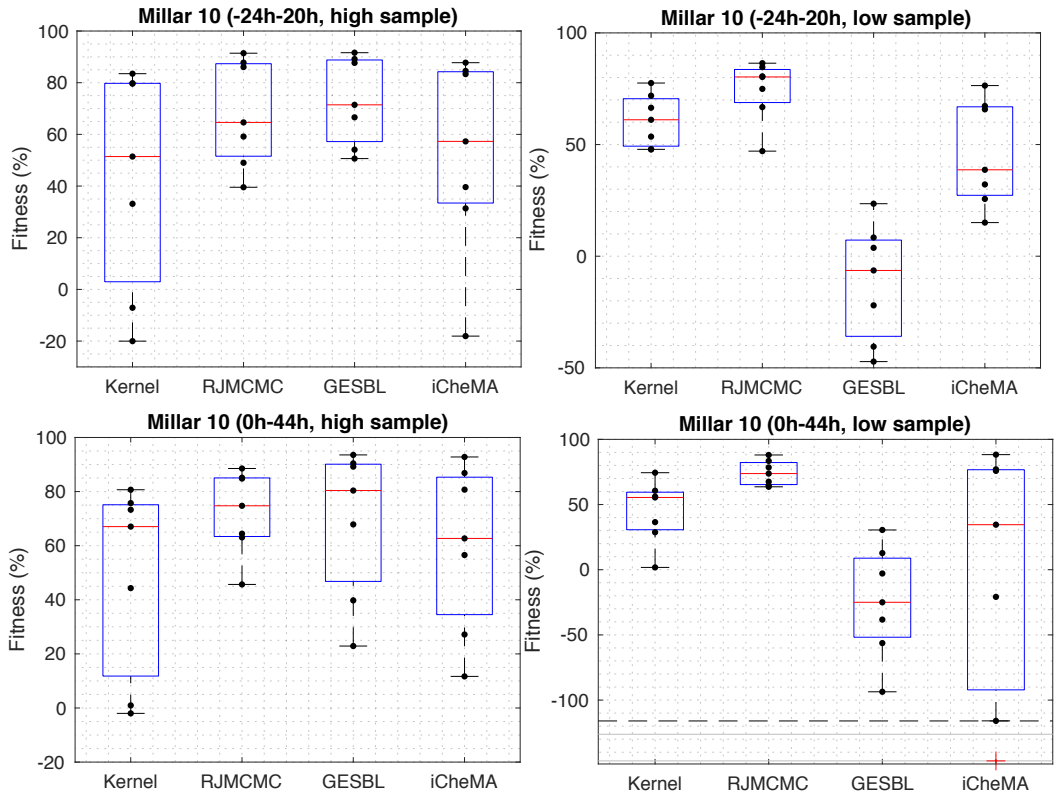


Figure 7.4.4: Prediction fitness of different methods. Each box plot shows the prediction fitness of all methods tested on a particular time window and sampling frequency. The average fitness of 7 genes is marked by the black points within the whiskers. The red lines indicate the median. The whiskers highlight the maximum and minimum data points that are not considered outliers. The bottom and top edges of the box represent the 25th and 75th percentiles, respectively. The outliers are plotted using the red '+' symbol.

Figure 7.4.4 shows the box plot of prediction fitness. In general, the fitness of the high sampling frequency is lower than the low sampling frequency because the models have to fit more data points with high sample. The fitness of time window -24h-20h is slightly worse than time window 0h-44h. Since time window -24h-20h contains the richest light transitions, its data are most difficult to fit. Among all the methods, RJMCMC presents the most reliable prediction. All points of RJMCMC are distributed compactly compared with the other methods, meaning that RJMCMC can provide reasonable prediction for most genes. The performance of GESBL is seriously degraded with the low sampling frequency. The fitness variance of iCheMA is very high, meaning that the models can only predict the dynamical behavior of a few genes. In particular, with time window 0h-44h and the low sampling frequency, the fitness of *PRR9* is below -100% (i.e. outlier in the plot). The kernel method is weak in prediction as the fitness variance of the high sampling frequency is the highest among all the methods for both time windows.

Simulations show that the performance of GESBL and iCheMA highly depends on the sampling frequency. That is because these two methods rely on the estimation of derivatives from data. With the low sampling frequency, derivatives cannot be recovered accurately by Gaussian regression, thus influencing the algorithm performance. With the high sampling frequency, Gaussian regression can effectively estimate derivatives from data so that the 'model power' of grey-box models in (7.4.4) and (7.4.1) is fully explored. In addition, since GESBL and iCheMA are focused on fitting the estimated derivatives, they are relatively weak in predicting new datasets.

We also compare the implementation time of all methods in Figure 7.4.5. Not surprisingly, the computational cost of iCheMA is extremely high as it conducts an exhaustive search of both network topology and internal dynamics (selection of basis functions) via numerical sampling. iCheMA requires around 3 hours to finish a single trial. It is remarkable that the computational cost of GESBL is very low. GESBL only takes a few seconds to complete the inference, which shows great advantages over the other methods. RJMCMC is less computationally intensive compared with iCheMA but still demands much more computational power than the remaining methods. The kernel method is the second computationally efficient approach. The computational cost of one-to-one grows quadratically with respect to the number of nodes. Since one-to-one adopts low order OE models to describe the network, its computational cost is much lower than RJMCMC and iCheMA.

Overall, RJMCMC is the most reliable method. It provides stable performance with different time windows and sampling frequencies. One-to-one also presents satisfactory results. More importantly, its computational cost scales with the size of the network. GESBL and iCheMA can only be used if high-resolution data are available. The kernel method is outperformed by the other methods.

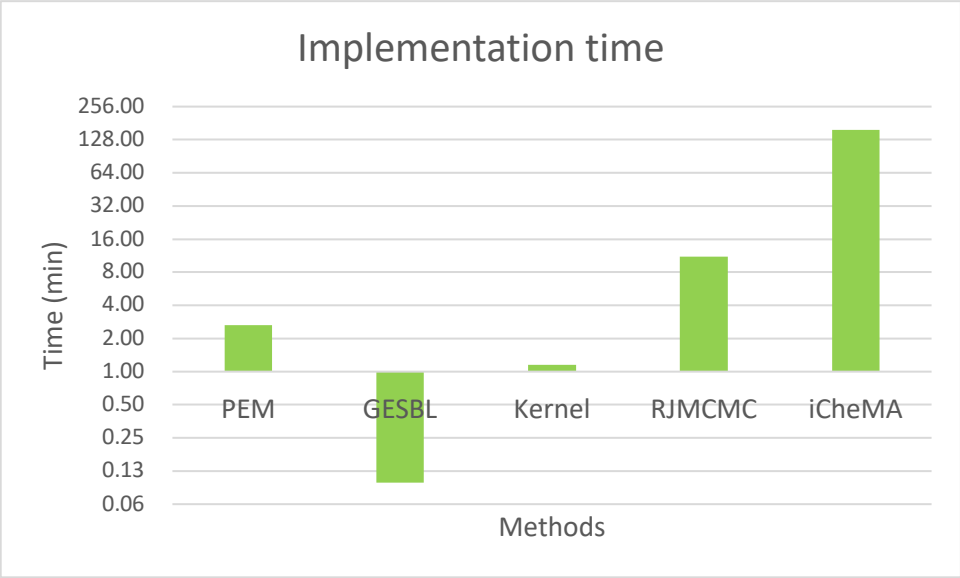


Figure 7.4.5: Implementation time of different methods. The red bars show the average time consumed by different methods to finish a single trial. For the purpose of a clear display, *y*-axis is scaled logarithmically with base 2.

7.5 Conclusion

This chapter conducts Monte Carlo simulations to verify the effectiveness of the proposed methods including one-to-one, GESBL, the kernel method and the RJMCMC approach. The proposed methods and a state-of-the-art approach are compared and tested on the synthesized models of the circadian clock. Different experimental conditions are considered including sampling frequency, light conditions and noise level.

Simulations show that high sampling frequency and light transitions dramatically improve the performance of the algorithms. RJMCMC and the kernel method are robust to truncation length of impulse responses. Hence, the problematic tuning is avoided. With the proposed linear models, RJMCMC, in general, outperforms all the other methods. By using grey-box nonlinear models, the accuracy of inference can be further improved. In this case, GESBL is superior to most of the proposed methods including the state-of-the-art method, iCheMA. The main drawback is that the algorithm performance highly depends on the estimation of derivatives. Hence, high-quality data are essential for GESBL.

The power of linear models is limited in describing large-scale biological networks that involve highly nonlinear interactions among nodes. Although the usage of grey-box nonlinear models has achieved certain success, model parameters must be linear with respect to basis functions. As a result, black-box nonlinear models are a promising model class to describe biological networks. Identification of nonlinear systems using the kernel method has been widely studied. Nevertheless, not much attention has been put on this area under the context of network inference.

Chapter 8.

Inference of the circadian Ca^{2+} signaling network

The circadian clock of *Arabidopsis* is a biological network providing regular 24 hour rhythms. It has attracted considerable research attention over the years (e.g. [225]–[228]). Learning its internal structure and underlying working mechanisms is fundamental to understanding many essential biological processes that are regulated by the circadian system. Mathematical modeling has been widely used to study the circadian clock, which has helped make considerable contributions to this research area [58], [59], [62], [63], [101], [103], [229]–[231].

An active topic relating to the circadian clock is the Ca^{2+} signaling network. Calcium is essential in many signaling pathways as a second messenger. For example, abscisic acid, salt stress, and cold shock and touch are sensed, transduced into Ca^{2+} signals and delivered through signaling pathways [232]. A great deal of research has been dedicated to learning the interaction between $[Ca^{2+}]_{cyt}$ and the circadian clock. Since *cca1-11* mutant abolishes the oscillation of $[Ca^{2+}]_{cyt}$ in constant white light, *CCA1* is believed to be one of the key components involved in $[Ca^{2+}]_{cyt}$ regulation [58]. In addition, this mutation also leads to undetectable $[Ca^{2+}]_{cyt}$ oscillation in constant blue and red light [103], which indicates that *CCA1* dominates $[Ca^{2+}]_{cyt}$ regulation. Mathematical modeling shows that *CCA1* is not the only regulator of $[Ca^{2+}]_{cyt}$ and $[Ca^{2+}]_{cyt}$ regulation is light induced [103]. $[Ca^{2+}]_{cyt}$ regulation under the blue and red light pathways presents very different phenotypes. In particular, experiments show that nicotinamide inhibits the blue light pathway whereas the red light pathway is not influenced [103], suggesting these two light pathways are physiologically separable. Furthermore, mathematical modeling indicates that cross-talks exist between these two light pathways when they are both activated under white light. However, such interactions are highly complex and hardly interpreted by linear models [233].

Previous research in [58], [103], [233] was mainly focused on the feedforward regulation of $[Ca^{2+}]_{cyt}$ from the circadian clock. A heuristic strategy was applied to search for the clock genes responsible for $[Ca^{2+}]_{cyt}$ regulation from a small group of candidate genes. Such an approach becomes unreliable and highly inefficient as more clock genes are under consideration. In addition, the feedback from $[Ca^{2+}]_{cyt}$ to the circadian clock remains unknown.

This chapter applies the developed methods in this thesis to infer the circadian Ca^{2+} signaling network (including the feedforward and feedback loops of $[Ca^{2+}]_{cyt}$) of *Arabidopsis* using high-

resolution time series data. $[Ca^{2+}]_{cyt}$ and six main clock genes are under consideration. The signaling networks under constant white, blue and red light are inferred independently. The inferred networks are compared to locate the key clock genes that bridge the circadian clock and $[Ca^{2+}]_{cyt}$ in different light pathways. Dynamical models are used to describe the signaling network. Without prior knowledge to construct models, black-box linear models are adopted for network inference. One-to-one and RJMCMC as the top two methods according to the previous simulations are employed to identify the postulated models. System identification is purely based on time series data of the network.

Section 8.1 introduces background knowledge of the circadian clock. Section 8.2 discusses details of the inference procedure. Section 8.3, 8.4 and 8.5 presents the inferred networks under constant blue, red and white light, respectively. Section 8.6 compares the signaling networks of different light pathways. Section 8.7 concludes the whole chapter.

8.1 The circadian clock in plants

The circadian rhythm is defined as a subset of biological rhythms with a period around 24h per cycle. It has been observed in most eukaryotes and in many prokaryotes [234]. It is characterized by two main features. Firstly, the circadian rhythm persists under constant environmental conditions since it is endogenous and sustained. Secondly, it is temperature compensated [235]. Single cells can maintain circadian rhythms and oscillation phases can be changed by temperature cycles. Another environmental cue that can affect the phase is light. The process of rephrasing is called entrainment.

The circadian clock acts as a timekeeper to synchronize biological processes to daily rhythms from environment [231]. The circadian rhythm is governed by the circadian clock [234]. The circadian system consists of three main blocks: a self-sustained central oscillator, input pathways incorporating environmental signals (*Zeitgeber*) such as light and temperature, and output pathways that control various physiological processes. The very different responses of the clock to a light pulse at different times indicate the clock self-regulates its sensitivity to external stimulus [235]. Temperature works with light to signal seasonal changes and regulate seedling process. On the other hand, some clock components regulate a large number of downstream genes. Inputs can entrain the circadian clock while outputs can in turn control the sensitivity of input pathways. The circadian clock participates in multiple physiological processes containing growth, metabolism, and abiotic and biotic stresses through its output rhythms [103], [59].

8.2.1 Interlocking structure of the circadian clock

More than 20 genes have already been found to form the circadian clock in *Arabidopsis*. The interaction among these clock genes results in the circadian rhythm. The primary mechanism of the circadian clock is based on interconnected transcriptional and translational feedback loops [59], [236]. Circadian regulation requires both transcriptional and post-transcriptional control. The expression level of genes along with daily oscillation of proteins follows the circadian rhythm, which is the main evidence of circadian regulation. It is known that some components in the clock can regulate the transcription of other clock genes at different times through all day. As a result, these components are classified into four time phases: morning,

day, afternoon and evening. Morning and evening genes are characterized by promoter motifs, morning element (ME) and evening element (EE), respectively [237], [238].

The clock also depends on the PRRs family including *PRR9*, *PRR7*, *PRR5*, *PRR3* and *PRR1* whose mRNA peaks from morning to dusk in sequence and is positively regulated by *CCA1*/*LHY* [239]. PRR9, PRR7 and PRR5 act as repressors of *CCA1* and *LHY* during the day. The connectivity of these components is more complex than a cascade structure [240], [241].

GI and *ELF3* are also found to be essential clock genes. They interact with other clock-associated proteins like E3 ubiquitin-ligase COP1. ELF3 negatively regulates input pathways of light through gating the light sensitivity at different times of a day [242]. *GI* has a great impact on *ZTL* thus influencing the degradation of proteins TOC1 and PRR5. ELF3, ELF4 and LUX cooperate as main components within the central oscillator [231], [243], [244], [245].

Generally speaking, different phased components of the clock activate or repress each other leading to complicated network mechanisms of timing. In addition to transcriptional regulation, the circadian clock also influences post-transcription processes including alternatively-splicing and protein-protein interaction [223].

8.2.2 Input pathways

Entrainment is the mechanism of shifting phase of the clock in response to environmental cues to maintain the circadian rhythm. Light-dark cycles are one of the main entrainment signals to the clock which accompany daily oscillations of the clock. Phase advances before dawn, delays after dusk and persists during the subjective day [246]. Photoreception of light with different wave length is mediated by various channels. *PHYA*, *B*, *D* and *E* take charge of the red light sensing while *CRY1* and *2* are responsible for the blue light. The resulting impact of light entrainment on the clock also depends on fluence rate [247].

Clock functions are influenced by transduced light signals via photoreceptors. Light inputs enter the circadian clock via several clock loops [248], [249]. Light also participates in proteolysis of clock proteins [250].

PHY and *CRY* families are also outputs of the circadian clock as they have rhythmic oscillations in constant light and dark. *PHYC* peaks around dawn. *PHYD* and *PHYE*, *PHYB* and *CRY1*, and *PHYA* and *CRY2* are maximally expressed at ZT2, ZT6 and ZT10, respectively.

Temperature is another environmental cue that can entrain the circadian clock. However, effective entrainment of thermocycles requires at least 4℃ of temperature variation between the subjective day and night. Clock genes *PRR7* and *9* have been found involved in temperature entrainment [251], [252], [246]. *FBH1* is a key component that modulates warm temperature signals and clock responses [253].

Sensitivity of temperature of the circadian clock is much lower than that of light. The circadian clock is robust to temperature variation and it maintains accurate timing over a wide range of temperature. The mechanism of the clock that protects the circadian rhythm from temperature perturbation is called temperature compensation [254], [253].

8.2.3 Physiological control of the circadian clock

A variety of physiological and biochemical processes, containing photosynthesis, leaf movement, hypocotyl elongation, stomatal movement and circumnutation are under the control of the circadian clock through its output pathways [234]. Synchronization of these

mechanisms with the time of day optimizes the outputs leading to competitive advantages [58].

Rhythmic transcript abundance in steady-state is an essential output of the circadian clock [234]. These transcripts are involved in flowering, flavonoid synthesis, lignin synthesis, cell elongation, nitrogen fixation, carbon metabolism, mineral assimilation and photosynthesis [234].

Photoperiodic response pathway is another well-defined output pathway that controls the transition from vegetative growth to flowering [234]. *GI* has been identified as the main clock gene that induces flowering. GI along with SPI activates the circadian expression of *CO* which encodes a zinc-finger protein that is activated by red and blue light [255]. The peak in the abundance of CO occurs during the night under short days whereas it appears during the light period in long days [255], [256]. The production of active CO induces the expression of a putative kinase inhibitor, *FT*. *FT* and *SOC1* determine flowering time by activating the floral identity genes in the shoot apical meristem [257].

Circadian regulation with carbohydrate biochemistry in terms of photosynthesis, starch accumulation and feedback in return optimizes the absorption and utilization of solar energy by plant [258]. The mechanism allows plants to take advantage of regular environmental changes such as those arising from days and seasons, and also random changes, including clouds passing across the sun [258]. While light harvesting and photosynthetic CO_2 fixation are associated to circadian regulation, the nature of mechanisms that underpin the control is not clear [259]. That is because the interaction between the circadian clock and chloroplast where photosynthesis occurs is poorly known [259].

8.2.4 Calcium signals and the circadian system

Ca^{2+} is at a low concentration because of its cytotoxicity. The intracellular concentration of Ca^{2+} ($[Ca^{2+}]$) is controlled through the bi-directional reaction of influx due to the high electrochemical gradient across the plasma membrane and internal stores inside the endoplasmic reticulum (ER) or vacuole, and efflux and buffering which takes charge of mediating amplitude and recovering time of calcium signal. Many physiological processes generate calcium oscillations in response to biotic or abiotic stimuli based on the reactions above. $[Ca^{2+}]$ oscillations are sensed by calcium-dependent proteins to trigger transcriptional factors leading to corresponding responses [260]. External entry outside the cell and internal stores of Ca^{2+} are two main sources of signaling cytosolic-free Ca^{2+} ($[Ca^{2+}]_{cyt}$).

The alteration of the concentration of $[Ca^{2+}]_{cyt}$ passes important information that can be perceived by downstream targets to coordinate biological processes [261]. Transient elevations and short-period fluctuation of Ca^{2+} signaling can be induced by temporal stimulus such as light, temperature and oxidative stress [262], [263]. Even without perturbation, $[Ca^{2+}]_{cyt}$ still oscillates with a circadian rhythm in *Arabidopsis* [264], which is thought to be a firm evidence to support the hypothesis that the circadian clock participates the regulation of $[Ca^{2+}]_{cyt}$ through a feedforward loop. As $[Ca^{2+}]_{cyt}$ is associated to the transduction of environmental signals, it is also expected to affect the operation of the circadian clock, particularly via input pathways as a feedback. Nicotinamide is one of a few chemical substances which are able to interrupt the circadian oscillation of $[Ca^{2+}]_{cyt}$. It was found that the circadian oscillation of $[Ca^{2+}]_{cyt}$ results from oscillation in the cellular concentration of cyclic

adenosine diphosphate ribose (cADPR). cADPR is able to mediate Ca^{2+} release from the ER and vacuole [265]. cADPR is generated due to the enzymatic action of ADP-ribosyl cyclase (ADPRc) through the degradation of NAD with nicotinamide as a byproduct. This chemical reaction is reversible by excess nicotinamide so nicotinamide is an inhibitor of cADPR. *CCA1* represses both ADPRc activity and the circadian oscillation of $[Ca^{2+}]_{cyt}$ (Abdul-Awal unpublished). Additionally, nicotinamide increases the period of the circadian oscillator and outputs such as leaf movement [265].

The circadian Ca^{2+} signaling network involving $[Ca^{2+}]_{cyt}$, the circadian clock and light as main components presents complex dynamics which are reflected by the observed phenotype due to internal or external excitation. It was found that loss-of-function mutants of *CCA1* abolished rhythmic oscillation of $[Ca^{2+}]_{cyt}$ in constant light [266] whereas *cca1*-1 maintained circadian oscillations of *LHY*, *CAB2*, *CAT2* and *CCR2* in constant light [267]. On the contrary, loss-of-function mutant of *ELF3* abolishes circadian oscillations of $[Ca^{2+}]_{cyt}$, *CAB2* and *CCR2* in constant light [268], [266]. In addition, single mutants of *phyB* or *cry1* completely abolish circadian oscillations of $[Ca^{2+}]_{cyt}$ in constant red or blue light. However, *phyB cry1* double mutant only represses amplitude of circadian oscillation of $[Ca^{2+}]_{cyt}$ in constant light [266]. The above observations suggest that interactions of $[Ca^{2+}]_{cyt}$, the circadian clock and light are realized through several channels. These channels are capable of operating individually. Nevertheless, they are not physiologically isolated in the network and may regulate each other if certain conditions are satisfied.

8.2 Inference procedure

We focused on inferring the Ca^{2+} signaling network under different light conditions. Candidate clock genes under investigation included *CCA1*, *LHY*, *PRR7*, *PRR9*, *GI* and *TOC1*. Experiments were conducted under constant white, blue and red light, independently. The promotor activities of the clock genes were measured via luciferase luminesce. The concentration of $[Ca^{2+}]_{cyt}$ was measured as AEQUORIN (AEQ) bioluminescence. Plants were first grown for 9 days in light/dark cycles and then moved to constant light conditions. The data under constant light conditions were measured for inference (Figure 8.2.1-8.2.4). In particular, the luciferase luminesce data were sampled every 1 hour whereas the AEQUORIN (AEQ) bioluminescence data were sampled every 2 hours. To achieve consistent time resolution, $[Ca^{2+}]_{cyt}$ data were interpolated using cubic spline (i.e. function 'interp1' in Matlab) at inter-sample time points.

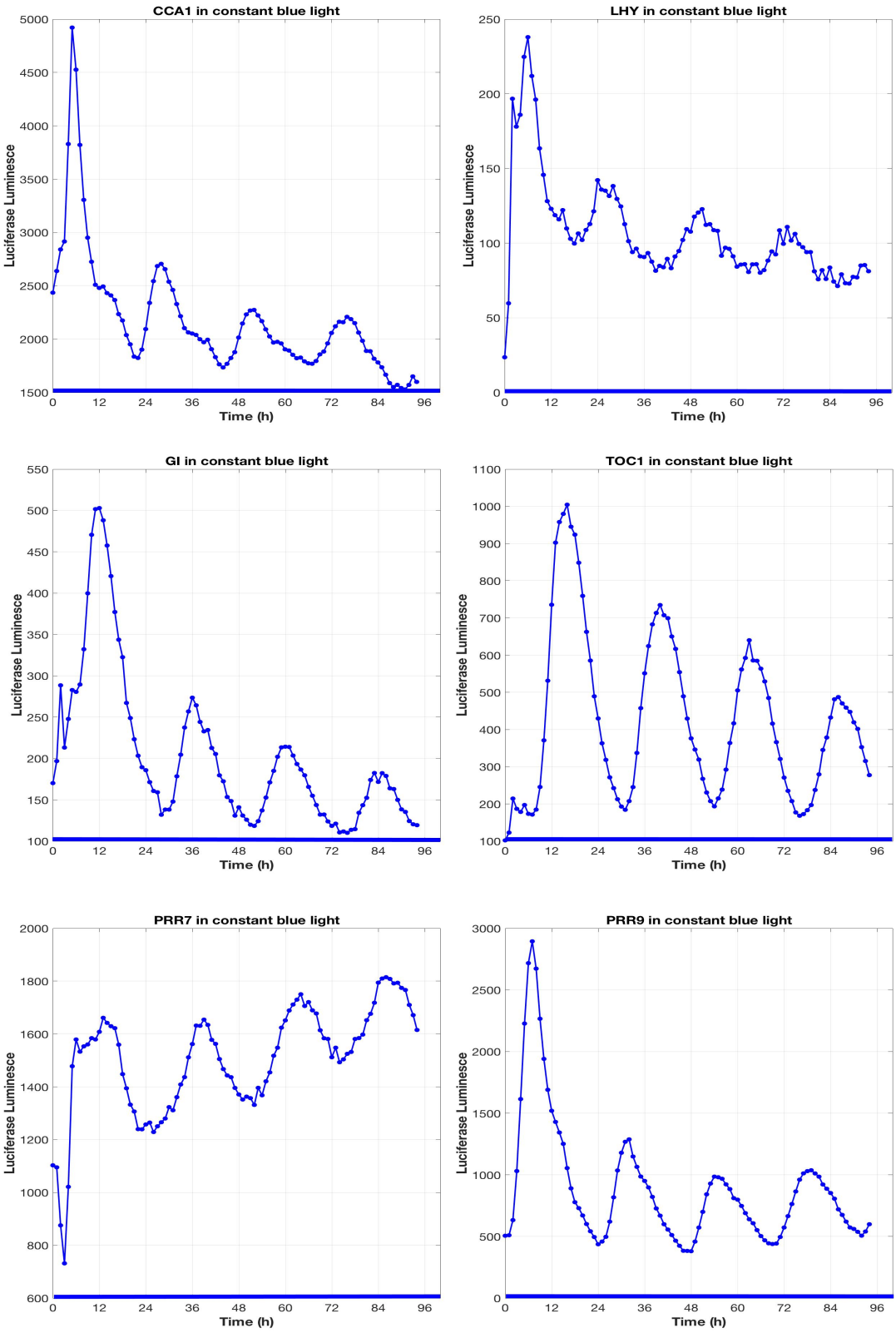


Figure 8.2.1: Promoter activity of clock genes *CCA1*, *LHY*, *PRR7*, *PRR9*, *GI* and *TOC1* in constant blue light. To avoid complex dynamics due to light transitions, the data of the first 24h were not used.
Data obtained by Dr Alberto Carignano

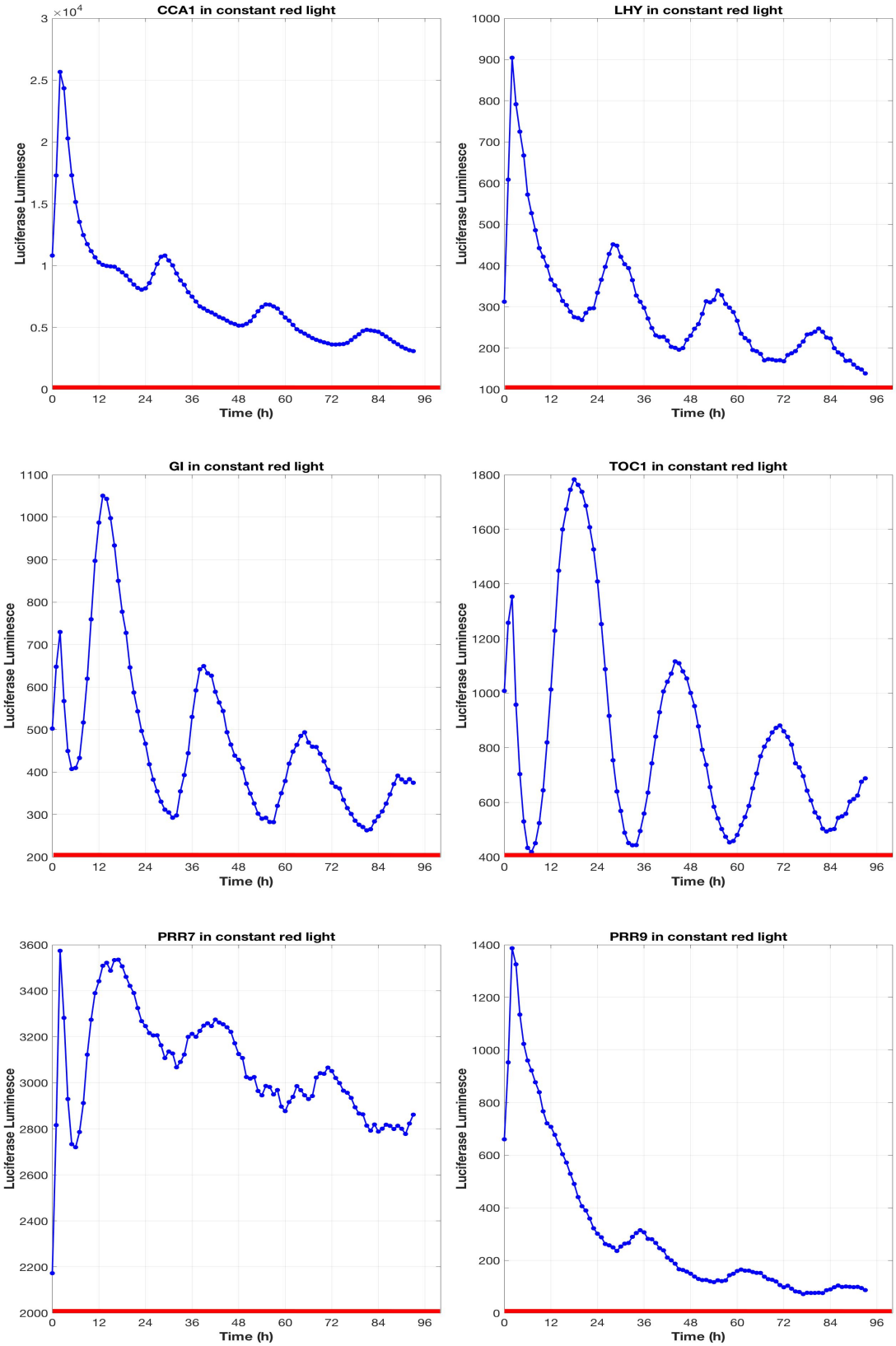


Figure 8.2.2: Promoter activity of clock genes *CCA1*, *LHY*, *PRR7*, *PRR9*, *GI* and *TOC1* in constant red light. To avoid complex dynamics due to light transitions, the data of the first 24h were not used.

Data obtained by Dr Alberto Carignano

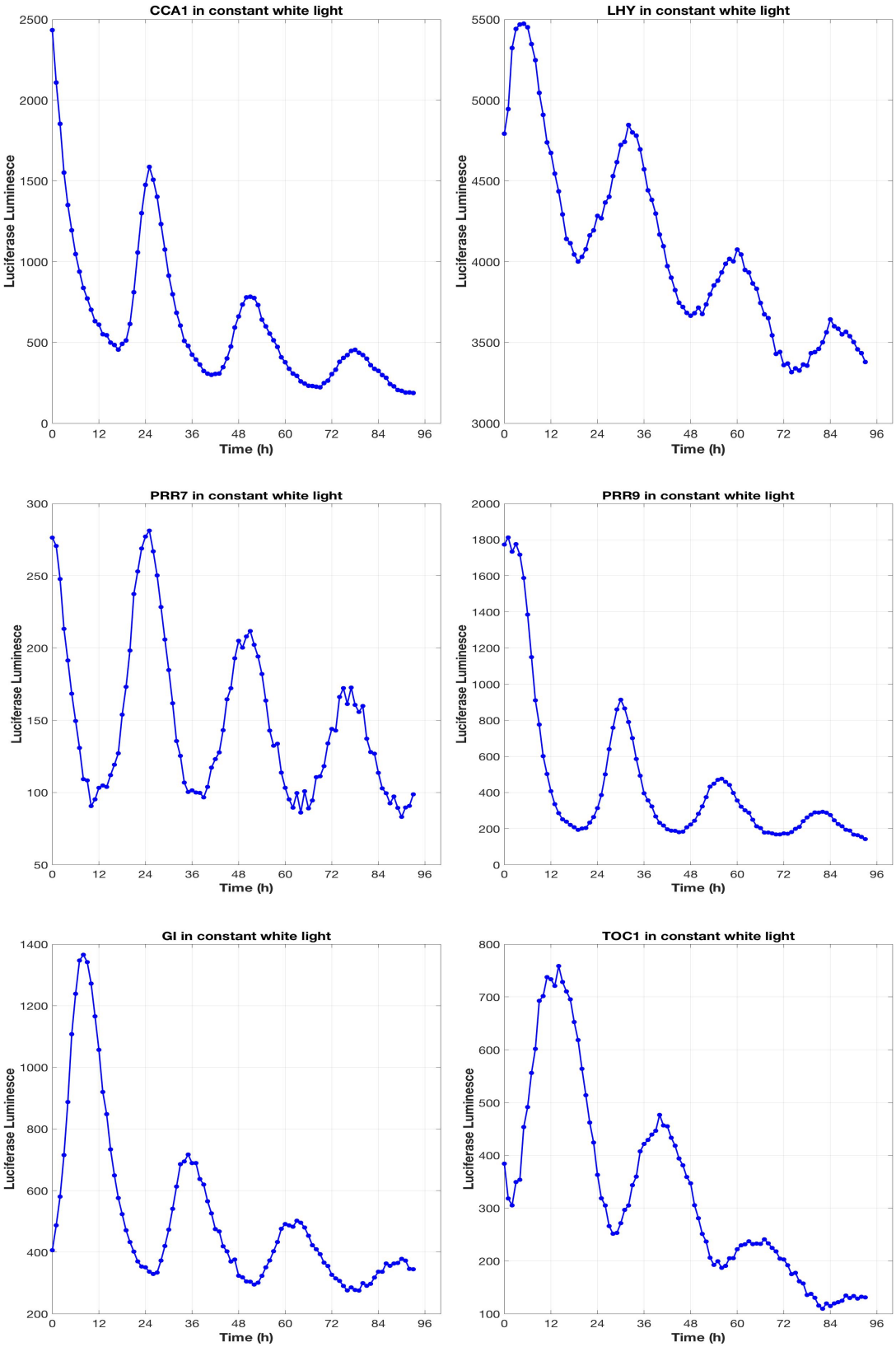


Figure 8.2.3: Promoter activity of clock genes *CCA1*, *LHY*, *PRR7*, *PRR9*, *GI* and *TOC1* in constant white light. To avoid complex dynamics due to light transitions, the data of the first 24h were not used.

Data obtained by Dr Alberto Carignano

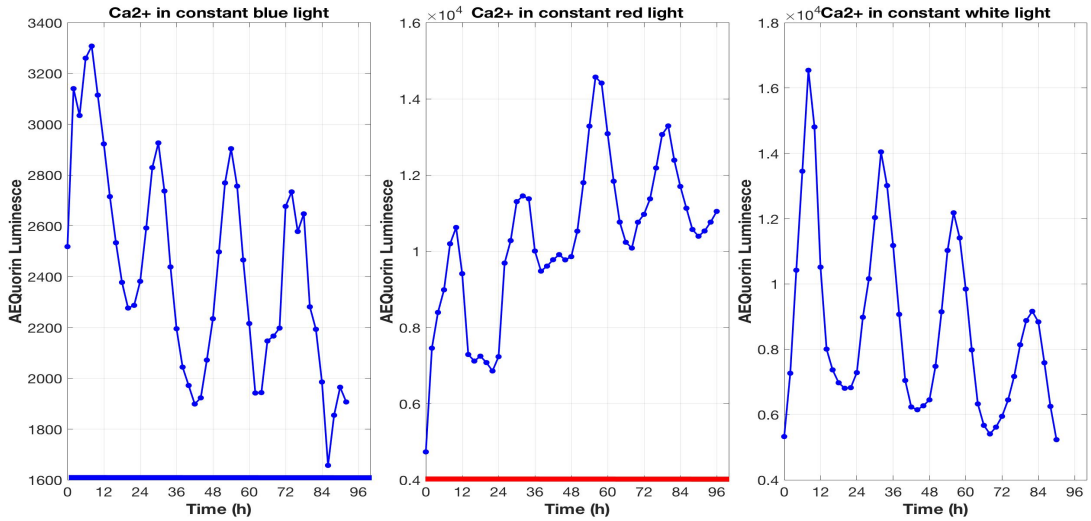


Figure 8.2.4: AEQ luminesce of $[Ca^{2+}]_{cyt}$ under different light conditions. To avoid complex dynamics due to light transitions, the data of the first 24h were not used.
Data obtained by Dr Alberto Carignano

The clock genes and $[Ca^{2+}]_{cyt}$ were treated as the nodes of the signaling network and light signals were the inputs. One-to-one and RJMCMC approaches were used to infer the network. The setup of these methods was the same with the previous simulations. Confidence measure was applied to reflect the reliability of inferred links. One-to-one used model fitness as the confidence measure whereas RJMCMC adopted the marginal posterior probability of inferred links. To avoid complex dynamics in response to the switch between different light pathways, the data of the first 24h were not used for inference. To guarantee reliability of inferred results, links whose confidence measure was under 70% were removed from inferred networks.

8.3 Blue light pathway of the Ca^{2+} signaling network

Figure 8.3.1 shows the inferred networks of the blue light pathway. One-to-one shows that $[Ca^{2+}]_{cyt}$ is only regulated by *CCA1* in the blue light pathway. It is known that *CCA1* regulates $[Ca^{2+}]_{cyt}$ via cADPR. Nicotinamide is able to repress cADPR, which cuts off the regulation pathway from *CCA1* to $[Ca^{2+}]_{cyt}$. Since $[Ca^{2+}]_{cyt}$ is only regulated by *CCA1* in the inferred network, we envision that nicotinamide abolishes the oscillation of $[Ca^{2+}]_{cyt}$ under constant blue light, which is consistent with the experimental observation. Although RJMCMC identifies additional regulators of $[Ca^{2+}]_{cyt}$ including *TOC1*, the regulation pathway from *CCA1* presents the highest confidence measure (89%).

One-to-one network suggests that $[Ca^{2+}]_{cyt}$ regulates *PRR9*, forming a feedback loop to the circadian clock whereas $[Ca^{2+}]_{cyt}$ regulates *PRR9* indirectly via *CCA1* and *TOC1* in RJMCMC network. In particular, *CCA1* and *TOC1* contribute equally to regulating *PRR9* (100% confidence inferred by RJMCMC), which may cause systematic fan-in error of one-to-one. *CCA1* and *TOC1* are regulated by similar genes in two inferred networks. The fact that one-to-one captures most gene regulators of *CCA1* and *TOC1* but misses $[Ca^{2+}]_{cyt}$ implies that these

genes provide similar regulations whereas $[Ca^{2+}]_{cyt}$ is involved in a very different way. $[Ca^{2+}]_{cyt}$ may have a significant impact on *CCA1* and *TOC1*. In addition, *CCA1* and *TOC1* are most active genes in both networks, suggesting they are the central core of the blue light pathway. Blue light has been reported to regulate $[Ca^{2+}]_{cyt}$ directly and indirectly via *CCA1* in [103]. RJMCMC network only reveals that *CCA1* is entrained by the blue light signal. Actually, inference results show that the probability of blue light directly affecting $[Ca^{2+}]_{cyt}$ is around 60%. Therefore, it is highly likely that the direct regulation from blue light involves complex dynamics and cannot be fully captured by linear models.

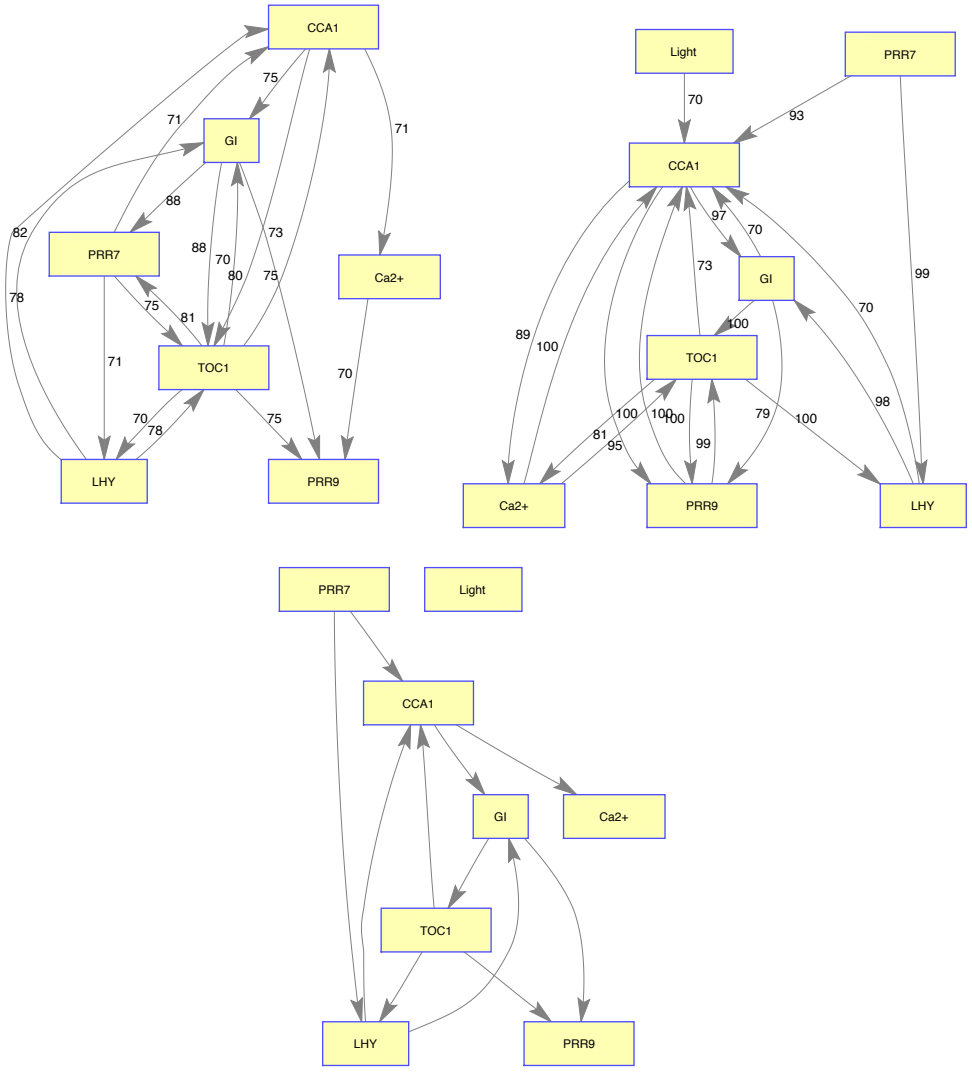


Figure 8.3.1: Ca^{2+} signaling network of the blue light pathway. Network (1,1) is inferred by one-to-one. Network (1,2) is inferred by RJMCMC. Network (2,1) is the common subnetwork shared by networks (1,1) and (1,2). Numbers in the graphs are the confidence measure of inferred links.

8.4 Red light pathway of the Ca^{2+} signaling network

Figure 8.4.1 displays inferred networks of the red light pathway. Compared with the blue

light pathway, the red light pathway is more complex. No clock genes are identified as the regulator of $[Ca^{2+}]_{cyt}$ in one-to-one network, meaning the regulations cannot be described by low order linear systems or the regulations involve multiple clock genes each of which regulates $[Ca^{2+}]_{cyt}$ in a unique way and cannot be replaced by other genes. Intriguingly, RJMCMC presents the opposite result that $[Ca^{2+}]_{cyt}$ is regulated by all clock genes except *LHY*, which verifies the possibility that $[Ca^{2+}]_{cyt}$ is regulated by multiple clock genes. This finding also provides a reasonable interpretation to the experimental observation that the oscillation of $[Ca^{2+}]_{cyt}$ is not inhibited by nicotinamide under constant red light. Although the regulation pathway from *CCA1* to $[Ca^{2+}]_{cyt}$ is blocked by nicotinamide, the circadian rhythm of $[Ca^{2+}]_{cyt}$ can still be reserved under the regulation of other clock genes.

One-to-one reveals that $[Ca^{2+}]_{cyt}$ regulates the clock via *CCA1, PRR9*, *LHY* and *TOC1* whereas RJMCMC shows no clock genes are regulated by $[Ca^{2+}]_{cyt}$. This means that the impact of $[Ca^{2+}]_{cyt}$ on the circadian clock can be, at least, well compensated by the interactions among clock genes under constant red light.

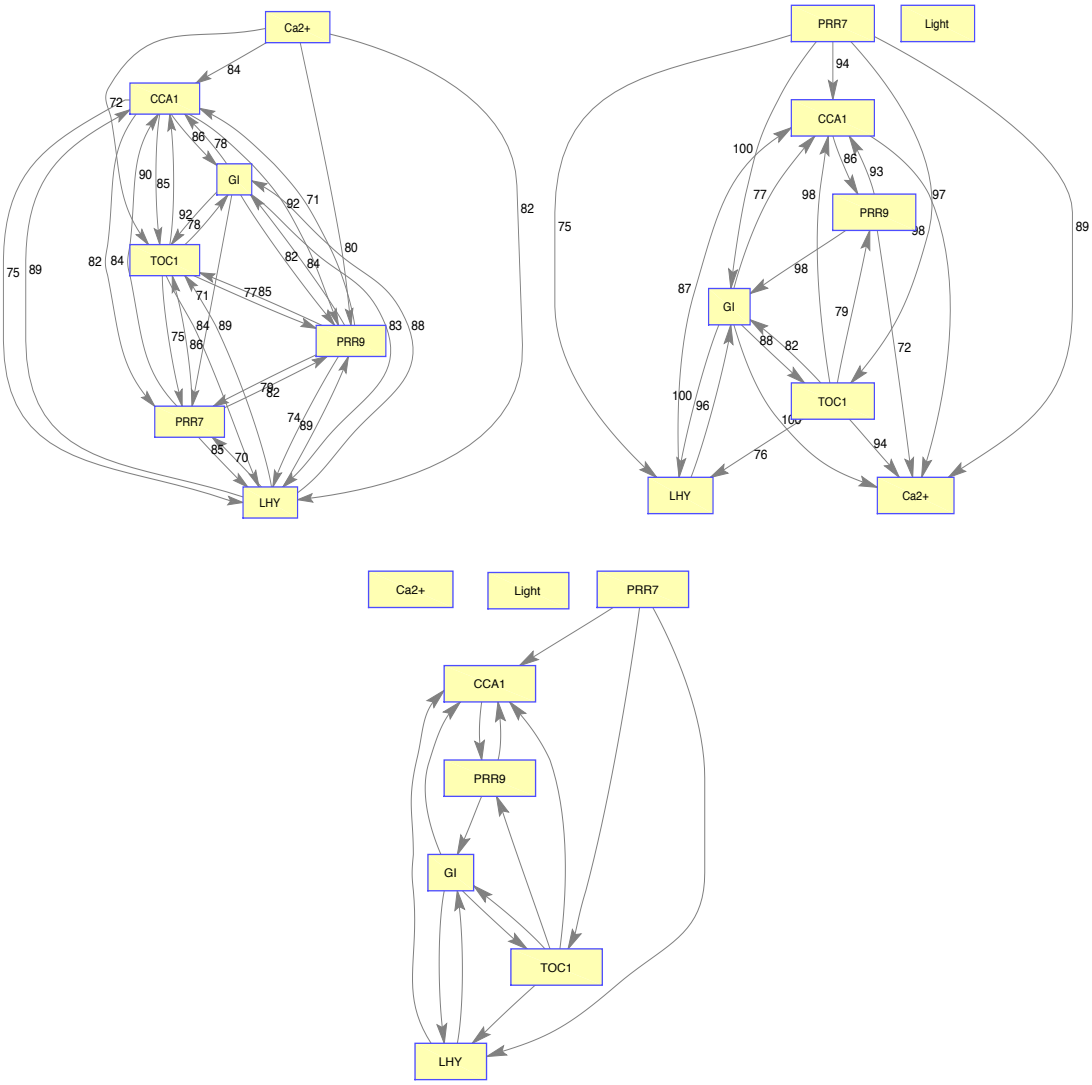


Figure 8.4.1: Ca^{2+} signaling network of the red light pathway. Network (1,1) is inferred by one-to-one. Network (1,2) is inferred by RJMCMC. Network (2,1) is the common subnetwork

shared by networks (1,1) and (1,2). Numbers in the graphs are the confidence measure of inferred links.

8.5 The Ca^{2+} signaling network under white light

Figure 8.5.1 shows the inferred networks under constant white light. Under constant white light, the blue and red light pathways are supposed to be both activated. One-to-one presents a heavily nested network whereas RJMCMC network is much sparser. Both networks show that $[Ca^{2+}]_{cyt}$ is regulated by *CCA1* and *PRR9* that have been previously identified as the key components in red and blue light pathways in [103]. *TOC1* and *PRR7* are identified as additional regulators of $[Ca^{2+}]_{cyt}$ by one-to-one whereas these two genes regulate $[Ca^{2+}]_{cyt}$ indirectly via *GI* in RJMCMC network.

One-to-one suggests $[Ca^{2+}]_{cyt}$ influences the circadian clock via *LHY* and *TOC1* whereas RJMCMC infers *GI* and *PRR9* as the targets of $[Ca^{2+}]_{cyt}$ forming a closed loop. The white light signal regulates *GI* and *PRR9* in RJMCMC network, which was also accepted to construct Millar models in [61], [62], [63].

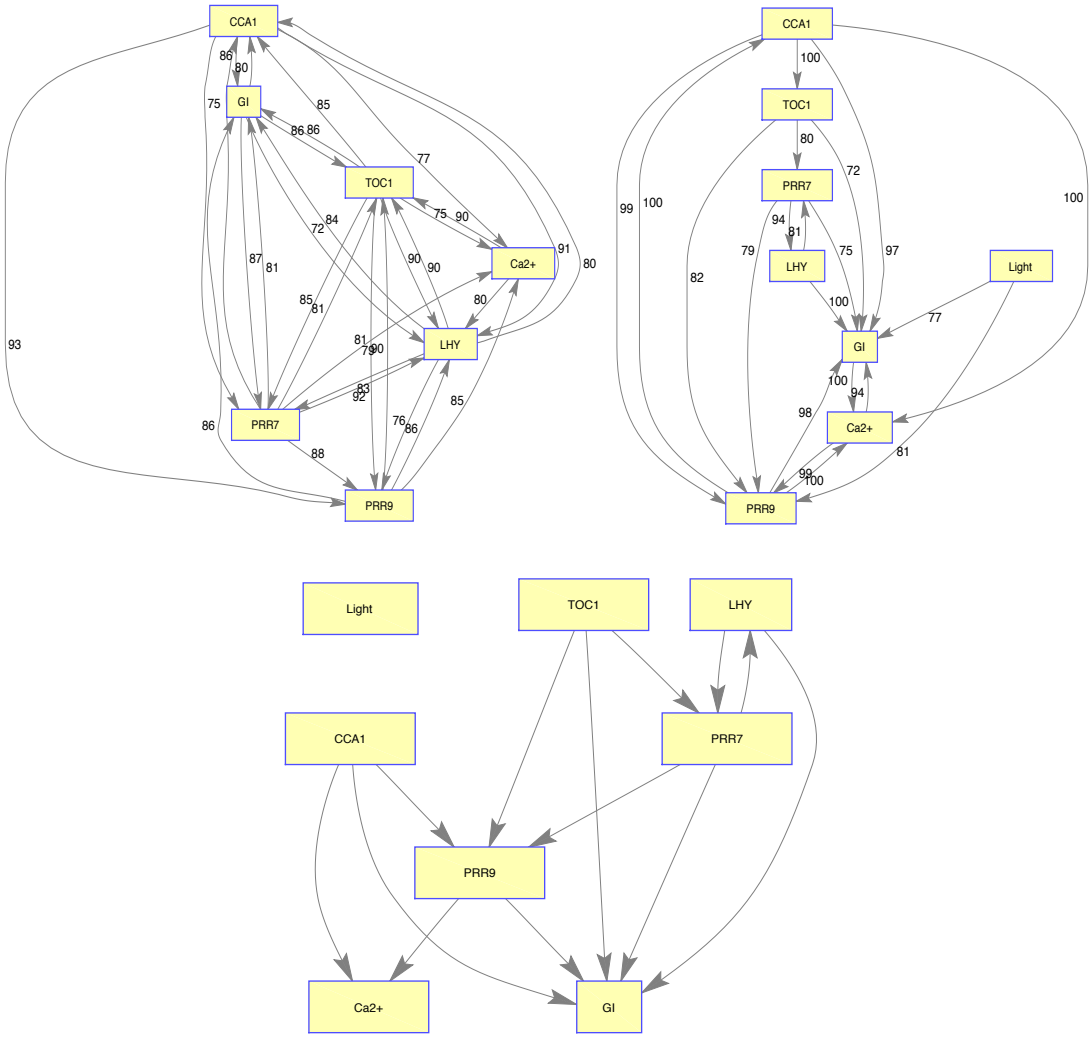


Figure 8.5.1: Ca^{2+} signaling network under constant white light. Network (1,1) is inferred by one-to-one. Network (1,2) is inferred by RJMCMC. Network (2,1) is the common subnetwork

shared by networks (1,1) and (1,2). Numbers in the graphs are the confidence measure of inferred links.

8.6 Interactions among light pathways

Figure 8.6.1 summarizes the interaction between the circadian clock and $[Ca^{2+}]_{cyt}$. We begin with the feedforward regulation from the circadian clock to $[Ca^{2+}]_{cyt}$. One-to-one networks show that *CCA1* regulates $[Ca^{2+}]_{cyt}$ under constant blue and white light conditions. *TOC1*, *PRR7* and *PRR9* only regulate $[Ca^{2+}]_{cyt}$ under constant white light. $[Ca^{2+}]_{cyt}$ is free-running under constant red light, which is highly unlikely since its oscillation sustains circadian rhythm that is mainly preserved by the six genes under consideration. As discussed, this result is probably caused by the systematic fan-in error of one-to-one. RJMCMC networks show that *CCA1* is responsible for $[Ca^{2+}]_{cyt}$ regulation in all light pathways, which is reasonable as it is the key component of the central oscillator. All the regulators of $[Ca^{2+}]_{cyt}$ under constant white light are also functional under constant red light. *GI* and *PRR9* are identified to regulate $[Ca^{2+}]_{cyt}$ under constant red and white light, among which *PRR9* was reported as the key element of the red light pathway [103]. In addition to the common regulators shared under different light conditions, the red and blue light pathways possess their unique regulators that are only functional under monotone light conditions. To be specific, *PRR7* and *TOC1* regulate $[Ca^{2+}]_{cyt}$ only under constant red and blue light, respectively. The inferred results indicate that red light behaves as an activator of the regulation pathways from the circadian clock to $[Ca^{2+}]_{cyt}$. In the absence of red light (i.e. under constant blue light), most regulation pathways are switched off. If the red and blue light pathways work independently, all regulators must be active under constant white light since both light pathways are activated. However, it is not the case according to the inferred results, meaning two light pathways interact with each other if they are both activated. In particular, one light pathway represses the unique regulator of the other. To be detailed, *PRR7* and *TOC1* are no longer the regulators of $[Ca^{2+}]_{cyt}$ under constant white light.

One-to-one and RJMCMC present very different feedback loops from $[Ca^{2+}]_{cyt}$ to the circadian clock. One-to-one indicates that the red light pathway contains all the feedback loops from $[Ca^{2+}]_{cyt}$ whereas $[Ca^{2+}]_{cyt}$ does not affect the circadian clock as suggested by RJMCMC. Inference results in above sections show that the signaling network is most active in the presence of red light, where nested interactions among the clock genes exist. Hence, one explanation of the disagreement between one-to-one and RJMCMC is that $[Ca^{2+}]_{cyt}$ regulates the target genes in a similar way as other clock genes. In other words, $[Ca^{2+}]_{cyt}$ is not the key factor that maintains the operation of the circadian clock in the presence of red light. $[Ca^{2+}]_{cyt}$ does not regulate the circadian clock in RJMCMC network because it can be replaced by other clock genes. It is worth noticing that one-to-one and RJMCMC both imply that $[Ca^{2+}]_{cyt}$ regulates different genes under constant blue and white light. In particular, no genes are regulated by $[Ca^{2+}]_{cyt}$ in all pathways. To this point, red light is more like a rail converter that switches the feedback loops of $[Ca^{2+}]_{cyt}$ rather than an activator. Meanwhile, red light might or might not activate additional feedback loops other than the ones of the blue and white light pathways.

Finally, *CCA1*, *TOC1* and *PRR9* are most significant components that build up the

communication between the circadian clock and $[Ca^{2+}]_{cyt}$. They are identified by both one-to-one and RJMCMC to be responsible for the cross-talks between the circadian clock and $[Ca^{2+}]_{cyt}$. Other clock genes are probably involved in the interaction indirectly via these genes.

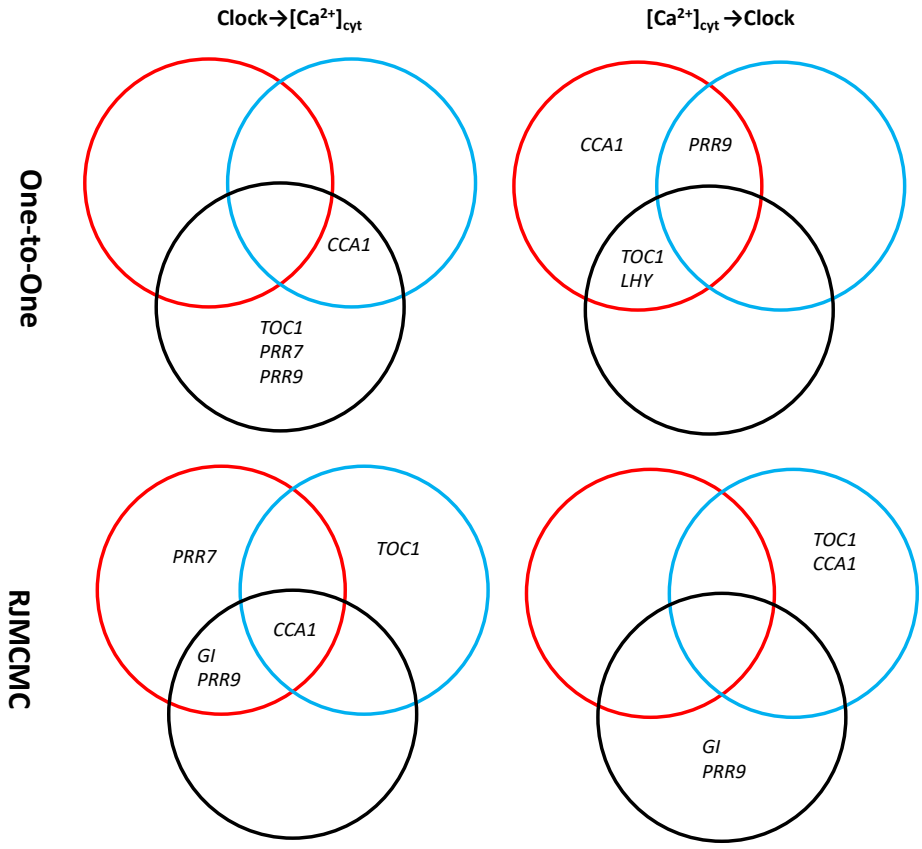


Figure 8.6.1: Ca^{2+} signaling network under different light conditions. The first and second rows display the networks inferred by one-to-one and RJMCMC, respectively. The first and second columns present the feedforward regulation to and feedback regulation from $[Ca^{2+}]_{cyt}$, respectively. The red, blue and white light pathways are denoted by red, blue and black circles, respectively.

8.7 Conclusion

This chapter applies novel methods to infer the Ca^{2+} signaling network of *Arabidopsis*. Interactions among the circadian clock and $[Ca^{2+}]_{cyt}$ are learned from high-resolution time series data. Six main clock genes and $[Ca^{2+}]_{cyt}$ as the nodes of the signaling network are under consideration. Light signals are the inputs of the network. Without prior knowledge to construct models, black-box linear models are used to describe the dynamical system. Since complex cross-talks exist between different light pathways and cannot be well interpreted by linear models according to the previous modeling [103], [233], the blue, red and white light pathways are inferred independently. One-to-one and RJMCMC approaches are applied to identify the postulated models. Inferred results are compared and analyzed to reveal the Ca^{2+} signaling network under different light conditions.

It is found that the Ca^{2+} signaling network is very different under different light conditions. Red light is fundamental to coordinate regulation pathways associated to $[Ca^{2+}]_{cyt}$. However, it plays different roles in controlling the feedforward and feedback loops of $[Ca^{2+}]_{cyt}$. Blue light is the activator of most regulators pathways from the circadian clock to $[Ca^{2+}]_{cyt}$. In the absence of red light (i.e. blue light condition), $[Ca^{2+}]_{cyt}$ is only regulated by a few clock genes. Most regulators of $[Ca^{2+}]_{cyt}$ are functional under constant white light. More importantly, when red and blue light pathways are both activated under white light, they interact with each other by repressing the activity of the unique regulator of the other light pathway. *PRR7* and *TOC1* that regulate $[Ca^{2+}]_{cyt}$ under constant red and blue light respectively no long serve as the regulators of $[Ca^{2+}]_{cyt}$ under constant white light.

Red light behaves like a switch that controls the feedback loops from $[Ca^{2+}]_{cyt}$ to the circadian clock. $[Ca^{2+}]_{cyt}$ regulates different genes under constant blue and white light. According to the inferred networks of one-to-one and RJMCMC, red light might or might not activate other regulation pathways in addition to the ones under constant blue and white light.

CCA1, *TOC1* and *PRR9* are most responsible for controlling the communication channel between the circadian clock and $[Ca^{2+}]_{cyt}$. They are active in both feedforward and feedback loops of $[Ca^{2+}]_{cyt}$. They also play important roles in multiple light pathways.

Current inference only considers six clock genes. It is highly possible that more clock genes are involved to bridge the circadian clock and $[Ca^{2+}]_{cyt}$. Therefore, future study should take other clock genes under consideration. In addition, since the interactions among different light pathways are nonlinear, nonlinear models are more appropriate to describe the signaling network. As a result, novel inference methods need to be developed to identify nonlinear models.

Chapter 9.

Conclusion

Biological networks are dynamical systems involving complex interactions among biological units. To fully understand a biological network, it is essential to learn causal relationships among biological units (network topology) and internal working mechanisms (internal dynamics). Mathematical modeling has been a powerful tool to systematically recover unknown biological networks from data. A variety of methods have been developed to infer biological networks. Some of these methods are focused on learning network topology where static probabilistic models are mostly applied to describe correlations among biological units. Many other methods also explore internal dynamics at the same time, employing dynamical models to track time evolution of biological systems. Dynamical models provide full knowledge of a biological network whose model structure reflects network topology and system properties represent internal dynamics. They can be used to simulate, analyze, predict and compare the behavior of biological networks under different conditions without repeatedly conducting experiments, thus saving considerable time and experimental cost. This thesis mainly focuses on developing novel dynamical model-based methods to infer biological networks. In comparison to many of the existing methods, the proposed methods in this thesis are especially dedicated in addressing issues related to network sparsity, system stability and unmeasured nodes. That is because many biological networks have much less connectivity than full-connected ones and these networks are intrinsically stable. In addition, many biological units are not measured in practical experiments. Therefore, solving these issues is imperative in real-world applications.

This thesis mainly applies black-box linear models and grey-box nonlinear models to describe biological networks. Construction of linear models requires no prior knowledge and hidden states can be easily encoded via transfer functions. By expressing linear models in a non-parametric way, a biological network can be inferred without knowing the number of unmeasured units. In particular, OE models, multivariable ARX models and DFSs are considered. Establishment of grey-box models incorporates prior knowledge of the target network and is based on physical laws. Hence, this type of model has explicit biological interpretations. Identification problems of the proposed models are formulated under the Bayesian framework. Prior distributions are introduced to impose model parsimony, network

sparsity and system stability. Bayesian techniques are applied to tackle intractable integrals of full Bayesian models. Empirical Bayes and MCMC sampling are used to analytically or stochastically relax the intractable problem. Overall, four inference methods are discussed, including one-to-one, GESBL, the kernel method and RJMCMC. Monte Carlo simulations show that these methods are very effective to infer networks that accurately fall in their postulated model classes.

The synthesized model (Millar 10) of the circadian system of *Arabidopsis* is adopted as the target network to further verify the methods. Models are simulated under different conditions to mimic practical experimental data such as microarray. Four proposed methods in this thesis are used to infer the network from data. These methods are also compared with a state-of-the-art method, iCheMA. Simulations show that GESBL equipped with grey-box nonlinear models and RJMCMC are superior to the other candidate methods. However, GESBL requires high-quality data while RJMCMC causes high computational cost.

A case study is conducted to learn the circadian Ca^{2+} signaling network of *Arabidopsis*. Six main clock genes are under consideration. These clock genes and $[Ca^{2+}]_{cyt}$ are nodes of the signaling network. Network inference is purely based on high-resolution time series data. Without prior knowledge, linear models are used to describe the network. RJMCMC and one-to-one methods are applied to infer the signaling network under different light conditions. The inferred networks indicate that the light pathways are very different.

9.1 Novel inference methods for sparse biological networks

Mathematical modeling has been widely used to infer biological networks. While many methods are focused on learning network topology, a full understanding of a biological network also requires knowledge of internal dynamics. Dynamical models are very powerful in describing the dynamical behavior of biological networks. With experimental data of the target network, dynamical models can be identified using system identification methods. In many system identification problems, model structure is not a main concern for the purpose of simulation, prediction and control. These problems are focused on identifying input-output relationships of the system. Under the context of network inference, determining model structure is essential because it reflects network topology. Therefore, many traditional system identification methods cannot be used to infer biological networks directly. This thesis presents novel methods to infer biological networks. Network topology and internal dynamics are both inferred from data with no or limited prior knowledge.

Biological networks are intrinsically stable and, in most cases, sparse. Hence, system stability and network sparsity are two major principles we consider in this thesis. Additionally, many biological species are not measured in practical experiments due to high expense, resulting in hidden nodes. Usually, the number of these hidden nodes is unknown, which further complicates inference problems. Finally, computational cost of inference methods is also a crucial matter in real-world applications. Diverse strategies are considered in this thesis to cope with these issues. Black-box linear models are used to describe biological networks so that hidden states can be encoded via transfer functions. The model is formulated in a non-parametric way without specifying the dimension of hidden states. Prior distributions are introduced to enforce system stability. The ARD technique is applied to promote network

sparsity. The connectivity of biological networks can be inferred in a pairwise manner to scale high the computational cost of large networks. Overall, we discuss four inference methods, including one-to-one, GESBL, the kernel method and RJMCMC.

One-to-one infers the connectivity of a network in a pairwise manner like correlation-based methods so that its computational cost scales with the size of the network. To characterize internal dynamics, one-to-one identifies an OE model for each ordered pair of nodes where the confidence of inferred links is measured via model fitness. This method can be used to analyze high-throughput data since only a simple model class with low model complexity is adopted to describe the target network. Overall, one-to-one provides more insights of the network than correlation-based methods without considerable increase in the demand of computational power and data. Theoretically, one-to-one is most suitable to infer networks consisting of chain links. The method is sensitive to fan-in and fan-out error. In addition, inputs of the network are not considered by this method, which causes systematic errors.

GESBL applies the ARD technique to impose network sparsity. In contrast to the one-to-one method, all links of a network are inferred simultaneously, thus effectively avoiding systematic errors. Linear and nonlinear multivariable ARX models are used to describe the network. Identification is formulated as a linear regression problem whose parameter vector is both group sparse due to network topology and element sparse in terms of model complexity. Group and element sparse Bayesian learning are combined to solve the regression problem. Empirical Bayes is applied to tackle the intractable integral of the Bayesian model. The computational cost of GESBL is light. Hence, it can be applied to infer large-scale networks.

The kernel method adopts DSFs to describe biological networks. Many real-world networks can be well approximated by DSFs so they have wide applicability in practical applications. DSFs encode information of hidden nodes so they are suitable to deal with incomplete datasets. By expressing models in a non-parametric way, DSFs can be proposed without prior knowledge of the number of hidden nodes. Gaussian process is used to impose system stability and network sparsity is promoted using the ARD technique. By applying empirical Bayes, internal dynamics of the models are optimized to interpret experimental data and model complexity (number of hidden nodes) is implicitly penalized for model parsimony. The kernel method has to solve a highly nonlinear optimization problem so the method is very sensitive to local optimal solutions.

RJMCMC also adopts DSFs to represent biological networks. In contrast to the kernel method, network topology is regarded as a random quantity. Instead of approximating the full Bayesian model analytically, RJMCMC explores the Bayesian model via numerical sampling. In particular, RJMCMC traverses freely across parameter subspaces of different dimensionality, each of which represents one possible topology. Therefore, RJMCMC encourages a global search of the sampling space, thus effectively avoiding many local maxima of the target distribution. Moreover, the confidence of the inferred topology can be evaluated via empirical distributions that converge asymptotically to the true distributions. By exploring the statistical properties of process noise, RJMCMC provides robust performance. The exhaustive search performed by RJMCMC causes high computational cost that is prohibitive for large-scale networks.

Simulations show that these developed methods are powerful in inferring networks that fall in their postulated model classes. These methods are further tested on the synthesized

circadian clock models as a case study to verify their effectiveness on inferring biological networks. Simulated data are collected under different sampling frequencies and time windows for network inference. It is found that high sampling frequency and rich light transitions greatly improve algorithm performance. With high sampling frequency, RJMCMC and GESBL equipped with grey-box nonlinear models are superior to the other candidate methods. However, GESBL presents poor performance given high-throughput data because the method relies on an accurate estimation of derivatives. One-to-one is the top-three method. It is more robust to sampling frequency than GESBL. In addition, RJMCMC and the kernel method are robust to truncation length of impulse responses. Hence, no tuning is required for their applications. A case study of the circadian Ca^{2+} signaling network using real experimental data further indicates the effectiveness of the proposed inference methods.

9.2 Red light controls the feedforward and feedback loops of $[Ca^{2+}]_{cyt}$

The signaling network of Ca^{2+} is inferred under constant blue, red and white light, independently. It is found that most feedforward pathways from the circadian clock to $[Ca^{2+}]_{cyt}$ are active under constant red light. Only a few clock genes regulate $[Ca^{2+}]_{cyt}$ under constant blue light, which makes the rhythmic oscillation of $[Ca^{2+}]_{cyt}$ less robust to external perturbations. Inferred regulation pathways under white light are all contained in the red light pathway, which further implies that red light acts as an activator of the feedforward loops to $[Ca^{2+}]_{cyt}$. More importantly, inference results suggest the blue and red light pathways interact with each other when they are both activated under white light. *PPR7* and *TOC1* that are identified as regulators of $[Ca^{2+}]_{cyt}$ under monotonic light conditions no longer regulate $[Ca^{2+}]_{cyt}$ under constant white light. A light signal represses certain regulators of $[Ca^{2+}]_{cyt}$ that belong to the other light pathway.

Red light plays very different roles in controlling the feedback loops from $[Ca^{2+}]_{cyt}$ to the circadian clock. Inferred results show that the blue and white light pathways share no common feedback loops. In addition, red light might or might not activate additional feedback loops. Therefore, red light behaves like a switch that swaps the feedback loops of blue light with white light.

CCA1, *TOC1* and *PRR9* are identified to be the most important genes involved in the cross-talks between the circadian clock and $[Ca^{2+}]_{cyt}$. It is highly likely that *CCA1* regulates $[Ca^{2+}]_{cyt}$ in all light pathways. $[Ca^{2+}]_{cyt}$ influences the circadian clock mainly through *TOC1* and *PRR9*.

Since only a few clock genes are investigated, it is possible that other clock genes participate in the interaction in a direct or indirect way. Therefore, it is necessary to consider more clock genes during inference. In addition, to learn the interaction among different light pathways, experiments should be conducted under various light conditions. Since previous modeling indicates that the interaction among light pathways is highly nonlinear, it is better to resort to nonlinear models for network inference.

9.3 Future work

9.3.1 Computational cost

Many biological networks have a large number of nodes and highly complex internal dynamics. To infer these networks, a large amount of data is required. Analyzing these data demands high computational power. Hence, for the purpose of implementation, the computational cost of inference methods must be scalable to the size of dataset. One-to-one is developed under this consideration where connectivity of a biological network is inferred in a pairwise manner. In addition, ADMM method is used to decompose the optimization problem of GEBSL into small sub-problems. Although RJMCMC has been shown to be a powerful method, the computational cost of RJMCMC is prohibitive for large-scale networks.

Manipulation of high-dimensional covariance matrices accounts for the most part of the computational cost in RJMCMC sampling loops. In particular, calculation of matrix determinant and inversion is the bottleneck of the algorithm. Therefore, more efficient algorithms to implement these operations are required. Low-rank approximations of covariance matrices are widely used to relieve the computational burden [74], [221]. There are also iterative methods such as fast Gaussian transform [269] and Fast Fourier Transform [270]. However, many of these methods are ill-suited for computing matrix determinant [221]. We have briefly introduced a state-of-the-art method in chapter 6 that can compute matrix inversion and determinant for general kernel functions with much lower cost compared to the standard methods. However, the main drawback is that the scaling of this method may not be optimal for some kernel functions, for example, oscillatory kernels without damping that are typically used to describe trajectories of rhythmic biological networks. Hence, further research on this topic is still necessary.

9.3.2 Models with measurement noise

Since measurement noise does not contribute to the internal dynamics of a network, this type of noise always degrades the performance of network inference methods. Additionally, in section 5.11, we find that system matrices of a DSF predictor are full if measurement noise exists regardless of the sparsity of the target network. Therefore, ARD in the current framework does not apply any more. Sparsity of networks cannot be imposed directly through system matrices. One way to deal with this problem is to consider measurement and process noise separately. In other words, the true trajectory of a state space model is treated as a latent random quantity apart from its noisy measurements.

The state space model with measurement noise for the target network is expressed as:

(9.3.1)

$$x(t+1) = Ax(t) + B_u u(t) + B_e e(t)$$
$$y(t) = Cx(t) + v(t)$$

where $x \in \mathbb{R}^n$ are states of the system, $u \in \mathbb{R}^m$ denote the inputs, $y \in \mathbb{R}^p$ represent the measurements of states, $e \in \mathbb{R}^q$ are the process noise and $v \in \mathbb{R}^p$ are the measurement noise. These two types of noise are assumed to be i.i.d. Gaussian with zero mean and diagonal covariance matrices $E\{e(t)e'(t)\} = diag\{\sigma_{e1}^2, \dots, \sigma_{eq}^2\}$, $E\{v(t)v'(t)\} = diag\{\sigma_{v1}^2, \dots, \sigma_{vp}^2\}$, respectively. For the simplicity of demonstration, we assume $\sigma_e^2 = \sigma_{e1}^2 = \cdots = \sigma_{eq}^2$ and

$\sigma_v^2 = \sigma_{v1}^2 = \cdots = \sigma_{vp}^2$. Only p out of n states are measured whereas other states represent hidden nodes.

Following the same procedure in chapter 5, the DSF for (9.3.1) is:

(9.3.2)

$$X = QX + PU + HE$$
$$Y = X + V$$

where $X \in \mathbb{R}^p$ denote the true trajectory of the measured nodes and $Y \in \mathbb{R}^p$ represent the noisy measurements of X. Q, P and H are system matrices. H is assumed to be diagonal to guarantee identifiability of the network. In addition, qH is monic and minimum-phase. Since the network is sparse, Q and P are sparse matrices.

The DSF in (9.3.2) is rewritten as:

(9.3.3)

$$X = F_u U + F_y X + \hat{E}$$
$$Y = X + V$$

where

$$F_u = (qH)^{-1}P$$
$$F_y = I - (qH)^{-1}(I - Q)$$
$$\hat{E} = q^{-1}E$$

Clearly, by separating measurement and process noise, system matrices F_u and F_y of (9.3.3) retrieve sparsity. Therefore, we can again apply ARD to promote sparse network topologies like chapter 5 and 6.

By expressing model (9.3.3) in a non-parametric way and truncating impulse responses as chapter 5, the ith node of the network is described by the model below:

(9.3.4)

$$x_i(t) = \sum_{j=1}^{p}\sum_{k=1}^{M} h_{ij}^{y}(k)x_j(t-k) + \sum_{j=1}^{m}\sum_{k=1}^{M} h_{ij}^{u}(k)u_j(t-k) + e_i(t)$$
$$y_i(t) = x_i(t) + v_i(t)$$

where $h(k)$ denote impulse responses of transfer functions of system matrices.

Note that the likelihood function for each node of DSF (9.3.3) cannot be considered independently because the function depends on the unknown true trajectories of all nodes. Therefore, all links of the network must be inferred simultaneously. Assume time series data from time indices $-M+1$ to N for each node and input are collected for inference. We use Y_0 and X_0 to present the data of time indices before 0 (initial condition of (9.3.4)) while Y and X denote the data of time indices from 1 to N. To avoid heavy notation, we assume there are no inputs in the model (i.e. $m = 0$). The following formulation can be directly extended to the case with inputs. The likelihood function for model (9.3.3) is:

(9.3.5)

$$p(Y|X, X_0, \sigma_v^2) = \prod_{i=0}^{N-1} p(y(N-i)|x(N-i), \sigma_v^2)$$
$$= |2\pi\sigma_v^2|^{-\frac{pN}{2}} exp\left(-\frac{1}{2\sigma_v^2}\|Y - X\|_F^2\right)$$

$$p(X,X_0|W,\sigma_e^2,\sigma_v^2)=p(X_0|\sigma_v^2)\prod_{i=0}^{N-1}p(x(N-i)|x(-M+1:N-i-1),W,\sigma_e^2)$$
$$=|2\pi\sigma_v^2|^{-\frac{pM}{2}}exp\left(-\frac{1}{2\sigma_v^2}\|X_0-Y_0\|_F^2\right)$$
$$\times|2\pi\sigma_e^2|^{-\frac{pN}{2}}exp\left(-\frac{1}{2\sigma_e^2}\left\|X-\sum_{j=1}^{p}A_jW_j\right\|_F^2\right)$$

where $\|\cdot\|_F$ denotes Frobenius norm and

(9.3.6)

$$Y=\begin{bmatrix}y_1(N)&\cdots&y_p(N)\\ \vdots&\ddots&\vdots\\ y_1(1)&\cdots&y_p(1)\end{bmatrix},Y_0=\begin{bmatrix}y_1(0)&\cdots&y_p(0)\\ \vdots&\ddots&\vdots\\ y_1(-M+1)&\cdots&y_p(-M+1)\end{bmatrix}$$
$$X=\begin{bmatrix}x_1(N)&\cdots&x_p(N)\\ \vdots&\ddots&\vdots\\ x_1(1)&\cdots&x_p(1)\end{bmatrix},X_0=\begin{bmatrix}x_1(0)&\cdots&x_p(0)\\ \vdots&\ddots&\vdots\\ x_1(-M+1)&\cdots&x_p(-M+1)\end{bmatrix}$$
$$W=[W_1{}'\quad\cdots\quad W_p{}']',\ W_j=\begin{bmatrix}h_{1j}^y&\cdots&h_{pj}^y\end{bmatrix},\ W_j(:,i)=h_{ij}^y=\begin{bmatrix}h_{ij}^y(1)&\cdots&h_{ij}^y(M)\end{bmatrix}'$$
$$A=[A_1\quad\cdots\quad A_p],\ A_j=\begin{bmatrix}x_j(N-1)&\cdots&x_j(N-M)\\ \vdots&\ddots&\vdots\\ x_j(0)&\cdots&x_j(-M+1)\end{bmatrix}$$
$$\sigma_e^2=E\{e_j^2(t)\},\ \sigma_v^2=E\{v_j^2(t)\}$$

To construct a full Bayesian model, we introduce prior distributions for impulse responses. Impulse responses are assumed to be Gaussian processes with zero mean and covariance functions to be the stable spline kernel. Hyperpriors are assigned to the hyperparameters of the kernel in the same way as chapter 6. As a result, the full Bayesian model is:

(9.3.7)

$$p(X,X_0,W,\beta,\lambda,\sigma_e^2,\sigma_v^2|Y)\propto p(Y|X,X_0,\sigma_v^2)p(X,X_0|W,\sigma_e^2)p(W|\beta,\lambda)p(\beta,\lambda,\sigma_e^2,\sigma_v^2)$$
$$=|2\pi\sigma_v^2|^{-\frac{pN}{2}}exp\left(-\frac{1}{2\sigma_v^2}\|Y-X\|_F^2\right)$$
$$\times|2\pi\sigma_v^2|^{-\frac{pM}{2}}exp\left(-\frac{1}{2\sigma_v^2}\|X_0-Y_0\|_F^2\right)$$
$$\times|2\pi\sigma_e^2|^{-\frac{pN}{2}}exp\left(-\frac{1}{2\sigma_e^2}\|X-AW\|_F^2\right)$$
$$\times|2\pi C|^{-\frac{1}{2}}exp\left(-\frac{1}{2}vec(W)'C^{-1}vec(W)\right)$$
$$\times p(\beta,\lambda,\sigma_e^2,\sigma_v^2)$$

where $vec(\cdot)$ means to vectorize a matrix. $\beta\in\mathbb{R}^{p\times p}$ and $\lambda\in\mathbb{R}^{p\times p}$ are hyperparameters of kernel functions. In particular, $[\beta]_{ij}$ and $[\lambda]_{ij}$ are hyperparameters of kernel function $\bar{k}(t,s;\lambda_{ij},\beta_{ij})=\lambda_{ij}k_{ij}(t,s;\beta_{ij})$ for h_{ij}^y and:

$$C=blkdiag\{\bar{K}_{11},\bar{K}_{12},\dots,\bar{K}_{1p},\dots,\bar{K}_{p1},\bar{K}_{p2},\dots,\bar{K}_{pp}\}$$

$\bar{K}_{ij} = \lambda_{ij} K_{ij}$: covariance matrix for impulse response h_{ij}^{y}

By completing squares with respect to W, (9.3.7) becomes

(9.3.8)

$$\begin{aligned} p(X, X_0, W, \beta, \lambda, \sigma_e^2, \sigma_v^2 | Y) \propto & |2\pi\sigma_v^2|^{-\frac{pN}{2}} exp\left(-\frac{1}{2\sigma_v^2}\|Y - X\|_F^2\right) \\ & \times |2\pi\sigma_v^2|^{-\frac{pM}{2}} exp\left(-\frac{1}{2\sigma_v^2}\|X_0 - Y_0\|_F^2\right) \\ & \times |2\pi\sigma_e^2|^{-\frac{pN}{2}} exp\left[-\frac{1}{2\sigma_e^2}\sum_{i=1}^{p}(w_i - \mu_i)'\Sigma_i^{-1}(w_i - \mu_i)\right] \\ & \times \prod_{i=1}^{p}|2\pi K_i|^{-\frac{1}{2}} exp\left[-\frac{1}{2}x_i'(\sigma_e^2 I + \Phi\bar{K}_i\Phi')^{-1}x_i\right] \\ & \times p(\beta, \lambda, \sigma_e^2, \sigma_v^2) \end{aligned}$$

where

$$\bar{K}_i = blkdiag\{\bar{K}_{i1}, \bar{K}_{i2}, \dots, \bar{K}_{ip}\}$$

$$\Phi = \begin{bmatrix} x_1(N-1:N-M) & \cdots & x_p(N-1:N-M) \\ \vdots & \ddots & \vdots \\ x_1(0:-M+1) & \cdots & x_1(0:-M+1) \end{bmatrix}$$

$$x_i = [x_i(N) \quad \cdots \quad x_i(1)]'$$

$$w_i = \left[(h_{i1}^{y})' \quad \cdots \quad (h_{ip}^{y})'\right]'$$

$$\mu_i = \frac{1}{\sigma_e^2}\Sigma_i\Phi' x_i$$

$$\Sigma_i^{-1} = \frac{1}{\sigma_e^2}\Phi'\Phi + \bar{K}_i^{-1}$$

Impulse responses W can be integrated out from posterior distribution (9.3.8), producing the following marginal distribution:

(9.3.9)

$$\begin{aligned} p(X, X_0, \beta, \lambda, \sigma_e^2, \sigma_v^2 | Y) \propto & |2\pi\sigma_v^2|^{-\frac{pN}{2}} exp\left(-\frac{1}{2\sigma_v^2}\|Y - X\|_F^2\right) \\ & \times |2\pi\sigma_v^2|^{-\frac{pM}{2}} exp\left(-\frac{1}{2\sigma_v^2}\|X_0 - Y_0\|_F^2\right) \\ & \times \prod_{i=1}^{p}|\sigma_e^2 I + \Phi K_i\Phi'|^{-\frac{1}{2}} exp\left[-\frac{1}{2}x_i'(\sigma_e^2 I + \Phi\bar{K}_i\Phi')^{-1}x_i\right] \\ & \times p(\beta, \lambda, \sigma_e^2, \sigma_v^2) \end{aligned}$$

We have discussed two frameworks in this thesis to identify DSFs without measurement noise: the kernel method (deterministic approximations) and RJMCMC (stochastic approximations). Here, we briefly show how these two approaches are applied to identify DSFs

with measurement noise.

1). The kernel method

Under the framework of the kernel method in chapter 5, conditional posterior distribution $p(W|\beta,\lambda,\sigma_e^2,\sigma_v^2,Y)$ is used to approximate the true marginal posterior distribution, $p(W|Y)$. Hyperparameters are optimized to minimize the gap between the approximate and the true distribution. The optimal hyperparameters are estimated by solving a Type II maximization problem:

(9.3.10)

$$(\hat{\beta},\hat{\lambda},\hat{\sigma}_e^2,\hat{\sigma}_v^2)=\arg\max_{\beta,\lambda,\sigma_e^2,\sigma_v^2} p(Y|\beta,\lambda,\sigma_e^2,\sigma_v^2)$$

Problem (9.3.10) cannot be solved directly since $p(Y|\beta,\lambda,\sigma_e^2,\sigma_v^2)$ is intractable. Nevertheless, $p(Y|\beta,\lambda,\sigma_e^2,\sigma_v^2)$ can be achieved by marginalizing W, X and X_0 out from $p(Y,X,X_0,W|\beta,\lambda,\sigma_e^2,\sigma_v^2)$ where $p(Y,X,X_0,W|\beta,\lambda,\sigma_e^2,\sigma_v^2)$ has a closed-form expression according to (9.3.7). Therefore, the EM algorithm is used to solve (9.3.10), treating W, X and X_0 as latent random quantities.

To further simplify notation, let $\theta=(\beta,\lambda,\sigma_e^2,\sigma_v^2)$ and $Z=(W,X,X_0)$. In the E step, function $Q(\theta|\theta^{t-1})=E_{Z|Y,\theta^{t-1}}[\log p(Y,Z|\theta)]$ of the current iteration is calculated using θ^{t-1} from the last iteration. Note that this expectation cannot be calculated analytically. Hence, a MCMC sampling framework (e.g. Gibbs sampling) is used in the E step to estimate $Q(\theta|\theta^{t-1})$, which results in a Monte Carlo Expectation Maximization (MCEM) algorithm [179]. With samples $\{Z_i^t, i=1,2,\dots,T\}$ drawn from $p(Z|Y,\theta^{t-1})$ in iteration t, $Q(\theta|\theta^{t-1})$ is approximated as:

(9.3.11)

$$\begin{aligned}Q(\theta|\theta^{t-1}) &\approx \frac{1}{T}\sum_{i=1}^{T}\log p(Y,Z_i^t|\theta)\\ &=\frac{1}{T}\sum_{i=1}^{T}L_1\left(Y,Z_i^t\middle|\sigma_v^2\right)+L_2\left(Y,Z_i^t\middle|\sigma_e^2\right)+L_3\left(Y,Z_i^t\middle|\beta,\lambda\right)\end{aligned}$$

where

$$\begin{aligned}L_1\left(Y,Z_i^t\middle|\sigma_v^2\right)&=-\frac{p(M+N)}{2}\log\sigma_v^2-\frac{1}{2\sigma_v^2}\left(\left\|Y-X_i^t\right\|_F^2+\left\|(X_0)_i^t-Y_0\right\|_F^2\right)\\ L_2\left(Y,Z_i^t\middle|\sigma_e^2\right)&=-\frac{pN}{2}\log\sigma_e^2-\frac{1}{2\sigma_e^2}\left\|X_i^t-A_i^tW_i^t\right\|_F^2\\ L_3\left(Y,Z_i^t\middle|\beta,\lambda\right)&=-\frac{1}{2}\left[\log|C|+vec(W_i^t)'C^{-1}vec(W_i^t)\right]\end{aligned}$$

In the M step, $\theta^t=\arg\max_\theta Q(\theta|\theta^{t-1})$ is solved. According to (9.3.11), hyperparameters can be estimated independently. The optimal σ_v^2 and σ_e^2 have closed-form solutions: $(\sigma_v^2)^t=\frac{1}{p(M+N)T}\sum_{i=1}^{T}\left\|Y-X_i^t\right\|_F^2+\left\|(X_0)_i^t-Y_0\right\|_F^2$ and $(\sigma_e^2)^t=\frac{1}{pNT}\sum_{i=1}^{T}\left\|X_i^t-A_i^tW_i^t\right\|_F^2$.

Optimizing β and λ can be decomposed into small sub-problems.

(9.3.12)

$$\left(\lambda_{ij}^t, \beta_{ij}^t\right) = \arg\min_{\lambda_{ij},\beta_{ij}} \frac{1}{T}\sum_{q=1}^{T} \log\left|\lambda_{ij}K_{ij}\right| + {h_q^t}'\left(\lambda_{ij}K_{ij}\right)^{-1}h_q^t$$

where h_q^t is the qth sample of h_{ij}^y from $\{Z_k^t, k = 1,2,\dots,T\}$.

The optimal solution for λ_{ij}^t depends on β_{ij}^t as:

(9.3.13)

$$\lambda_{ij}^t = \frac{1}{MT}\sum_{q=1}^{T} {h_q^t}'\left(K_{ij}\right)^{-1}h_q^t$$

Inserting (9.2.13) back into (9.3.12), we have:

(9.3.14)

$$\beta_{ij}^t = \arg\min_{\beta_{ij}} M\log\sum_{q=1}^{T} {h_q^t}'\left(K_{ij}\right)^{-1}h_q^t + \log\left|K_{ij}\right|$$

With the optimized hyperparameters, impulse responses W are estimated as the mean of the conditional posterior distribution: $\widehat{W} = E\{p(W|\beta,\lambda,\sigma_e^2,\sigma_v^2,Y)\}$. However, calculating $\widehat{W}$ requires to perform a high-dimensional integral (i.e. $\widehat{W} = \int Wp(X,X_0,W|\beta,\lambda,\sigma_e^2,\sigma_v^2,Y)dWdXdX_0$), which has no analytical solution. In particular, distribution $p(X,X_0,W|\beta,\lambda,\sigma_e^2,\sigma_v^2,Y)$ is only known up to a constant. Additionally, $p(W|\beta,\lambda,\sigma_e^2,\sigma_v^2,Y)$ is non-Gaussian after marginalizing out X and X_0. As a result, numerical sampling methods such as MCMC are used to draw samples from $p(X,X_0,W|\beta,\lambda,\sigma_e^2,\sigma_v^2,Y)$. With the samples of W, $E\{p(W|\beta,\lambda,\sigma_e^2,\sigma_v^2,Y)\}$ can be estimated using the empirical distribution.

To conclude, the algorithm to identify (9.3.3) using the kernel method is as follows:

Algorithm 1 The kernel method equipped with MCEM

1: ***Initialization***: set $((\sigma_v^2)^0,(\sigma_e^2)^0,\beta^0,\lambda^0)$

2: ***For*** $t = 1{:}\,MAX$ do

3: Draw samples $\{(W_i^t,X_i^t,(X_0)_i^t), i = 1,2,\dots,T\}$ from $p(W,X,X_0|(\sigma_v^2)^{t-1},(\sigma_e^2)^{t-1},\beta^{t-1},\lambda^{t-1},Y)$

4: Update σ_v^2 as

$$(\sigma_v^2)^t = \frac{1}{p(M+N)T}\sum_{i=1}^{T}\left\|Y - X_i^t\right\|_F^2 + \left\|(X_0)_i^t - Y_0\right\|_F^2$$

5: Update σ_e^2 as

$$(\sigma_e^2)^t = \frac{1}{pNT}\sum_{i=1}^{T}\left\|X_i^t - A_i^tW_i^t\right\|_F^2$$

6: Update β independently by solving

$$\beta_{ij}^{t} = \arg\min_{\lambda_{ij}} M\log \sum_{q=1}^{T} h_q^{t\prime}\left(K_{ij}\right)^{-1} h_q^{t} + \log\left|K_{ij}\right|$$

7: Calculate K_{ij} using β_{ij}^{t} and update λ independently as

$$\lambda_{ij}^{t} = \frac{1}{MT}\sum_{q=1}^{T} h_q^{t\prime}\left(K_{ij}\right)^{-1} h_q^{t}$$

8: ***End for***

9: Draw samples $\{W_i, i = 1,2,\dots,T\}$ from $p(W, X, X_0|\hat{\sigma}_v^2, \hat{\sigma}_e^2, \hat{\beta}, \hat{\lambda}, Y)$

10: Calculate $\widehat{W} = \frac{1}{T}\sum_{i=1}^{T} W_i$

The performance of MCEM highly depends on whether the sampler of the E step is well-designed. In general, there is no guarantee that the generated sequence will converge to the stationary points of the marginal likelihood in (9.3.10). Nevertheless, it is commonly agreed that the number of samples, T should increase with the number of iterations like simulated annealing methods [106].

2). RJMCMC approach

To apply RJMCMC, network topology is also treated as a random quantity and assigned with a prior distribution (refer to chapter 6). Therefore, the resulting full Bayesian model is:

(9.3.15)

$$\begin{aligned}
&p(X, X_0, W_k, \beta_k, \lambda_k, k, \alpha, \sigma_e^2, \sigma_v^2|Y)\\
&\propto p(Y|X, X_0, \sigma_v^2)p(X, X_0|W_k, k, \sigma_e^2)p(W_k|\beta_k, \lambda_k, k)p(\beta_k, \lambda_k|k)p(k|\alpha)p(\sigma_e^2, \sigma_v^2, \alpha)\\
&= |2\pi\sigma_v^2|^{-\frac{pN}{2}} exp\left(-\frac{1}{2\sigma_v^2}\|Y - X\|_F^2\right)\\
&\times |2\pi\sigma_v^2|^{-\frac{pM}{2}} exp\left(-\frac{1}{2\sigma_v^2}\|X_0 - Y_0\|_F^2\right)\\
&\times |2\pi\sigma_e^2|^{-\frac{pN}{2}} exp\left(-\frac{1}{2\sigma_e^2}\|X - A_k W_k\|_F^2\right)\\
&\times |2\pi C_k|^{-\frac{1}{2}} exp\left(-\frac{1}{2} vec(W_k)' C_k^{-1} vec(W_k)\right)\\
&\times p(k, \alpha, \beta_k, \lambda_k, \sigma_e^2, \sigma_v^2)
\end{aligned}$$

where k is the index for network topology. The structure of the quantities labelled with subscript k changes accordingly with respect to network topology.

In contrast to the kernel method, marginal distribution $p(W|Y)$ is not approximated analytically using $p(W|\beta, \lambda, \sigma_e^2, \sigma_v^2, Y)$. Rather, $p(W|Y)$ is evaluated via numerical sampling. Here, we design a sampler for the full Bayesian model in (9.3.15) following the same procedure in chapter 6.

To begin with, a basic Gibbs sampler is used to sample the model (Sampler 1). Random quantities whose structure is related to network topology k are grouped together.

Sampler 1 Block Gibbs sampler

1: Sample $p(X^{t+1}, X_0^{t+1}|W_k^t, \beta_k^t, \lambda_k^t, (\sigma_e^2)^t, (\sigma_v^2)^t, \alpha^t, k^t, Y)$

2: Sample $p(W_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1} | X^{t+1}, X_0^{t+1}, (\sigma_e^2)^t, (\sigma_v^2)^t, \alpha^t, Y)$
3: Sample $p((\sigma_v^2)^{t+1} | X^{t+1}, X_0^{t+1}, W_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1}, (\sigma_e^2)^t, \alpha^t, Y)$
4: Sample $p((\sigma_e^2)^{t+1} | X^{t+1}, X_0^{t+1}, W_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1}, (\sigma_v^2)^{t+1}, \alpha^t, Y)$
5: Sample $p(\alpha^{t+1} | X^{t+1}, X_0^{t+1}, W_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1}, (\sigma_v^2)^{t+1}, (\sigma_e^2)^{t+1}, Y)$

Conditional distributions for σ_v^2 and σ_e^2 in step 3 and 4 are Inverse-Gamma distributions. Hence, these two random variables can be sampled easily. The other sampling steps cannot be implemented directly due to the complexity of distributions. As a result, these steps are replaced by MH samplers. Consequently, we achieve a MH-within-Gibbs sampler (Sampler 2).

Sampler 2 MH-within-Gibbs sampler
1: Sample $p(X^{t+1}, X_0^{t+1} | W_k^t, \beta_k^t, \lambda_k^t, (\sigma_e^2)^t, (\sigma_v^2)^t, \alpha^t, k^t, Y)$ using MH algorithm
2: Sample $p(W_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1} | X^{t+1}, X_0^{t+1}, (\sigma_e^2)^t, (\sigma_v^2)^t, \alpha^t, Y)$ using MH algorithm
3: Sample $p((\sigma_v^2)^{t+1} | X^{t+1}, X_0^{t+1}, W_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1}, (\sigma_e^2)^t, \alpha^t, Y)$
4: Sample $p((\sigma_e^2)^{t+1} | X^{t+1}, X_0^{t+1}, W_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1}, (\sigma_v^2)^{t+1}, \alpha^t, Y)$
5: Sample $p(\alpha^{t+1} | X^{t+1}, X_0^{t+1}, W_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1}, (\sigma_v^2)^{t+1}, (\sigma_e^2)^{t+1}, Y)$ using MH algorithm

Step 2 in Sampler 2 requires sampling a series of high-dimensional quantities. To implement this step more efficiently, random quantity W_k^{t+1} is marginalized out from step 2 and sampled immediately in the next step according the rule of marginalization. Therefore, we achieve a MH-within-PCG sampler (Sampler 3).

Sampler 3 MH-within-PCG sampler
1: Sample $p(X^{t+1}, X_0^{t+1} | W_k^t, \beta_k^t, \lambda_k^t, (\sigma_e^2)^t, (\sigma_v^2)^t, \alpha^t, k^t, Y)$ using MH algorithm
2: Sample $p(\beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1} | X^{t+1}, X_0^{t+1}, (\sigma_e^2)^t, (\sigma_v^2)^t, \alpha^t, Y)$ using MH algorithm
3: Sample $p(W_k^{t+1} | X^{t+1}, X_0^{t+1}, , \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1}, (\sigma_e^2)^t, (\sigma_v^2)^t, \alpha^t, Y)$
4: Sample $p((\sigma_v^2)^{t+1} | X^{t+1}, X_0^{t+1}, W_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1}, (\sigma_e^2)^t, \alpha^t, Y)$
5: Sample $p((\sigma_e^2)^{t+1} | X^{t+1}, X_0^{t+1}, W_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1}, (\sigma_v^2)^{t+1}, \alpha^t, Y)$
6: Sample $p(\alpha^{t+1} | X^{t+1}, X_0^{t+1}, W_k^{t+1}, \beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1}, (\sigma_v^2)^{t+1}, (\sigma_e^2)^{t+1}, Y)$ using MH algorithm

Distribution $p(\beta_k^{t+1}, \lambda_k^{t+1}, k^{t+1} | X^{t+1}, X_0^{t+1}, (\sigma_e^2)^t, (\sigma_v^2)^t, \alpha^t, Y)$ in step 2 of Sampler 3 is known up to a constant according to (9.3.9). Therefore, step 2 is a feasible sampling step. Note that, the dimension of the sample space in step 2 is also a random quantity. Hence, RJMCMC is applied to draw samples. Step 3 can be implemented directly because W_k has a conditional Gaussian distribution.

Finally, Sampler 7 in chapter 6 can be used to implement steps from 2 to 6 of Sampler 3 above with minor modifications. Meanwhile, step 1 that samples the true trajectory of the system must be designed carefully.

To conclude, we have outlined the identification of DSF models with measurement noise under the frameworks of the kernel method and RJMCMC. By considering measurement and process noise separately, system matrices of the predictor retrieve sparsity due to sparse network topology. Hence, the ARD technique can be used to promote sparsity. Measurement

noise does not contribute to the internal dynamics of a network. While process noise helps excite the system, measurement noise twists the collected data thus interfering the reliability of inferred networks. The existence of measurement noise greatly complicates inference procedures. First, sub-networks that present regulations of each node cannot be inferred in parallel like the noise-free case, which reduces the efficiency of the developed algorithms. Second, the true trajectory of the system is regarded as a latent random quantity and sampled both in the kernel method and RJMCMC, which is essential to filter out measurement noise in the collected data. Sampling the true trajectory causes high computational cost that increases with the scale of a network and the size of datasets.

9.3.2 Application of continuous time models

This thesis mainly applies discrete time models to describe biological networks and presents corresponding system identification methods. Identification of continuous time models is briefly discussed in chapter 8. In particular, continuous time grey-box nonlinear models are identified using the estimated derivatives from data. In this case, a deterministic model (without process noise) is used to describe the target network and only measurement noise is considered. Since the Wiener process is nowhere differentiable [271], the derivative of the trajectory of a SDE model is not well defined. Hence, the derivative-based framework (i.e. GESBL and iCheMA) cannot be used to identify SDE models. With high sampling frequency, a SDE model can be well approximated by a discrete time stochastic model. Nevertheless, it is difficult to tell whether the sampling frequency is fast enough compared to the dynamics of an unknown biological network. On one hand, low sampling frequency causes system aliasing so that the resulting discrete time model no long represents the true dynamics and the corresponding network topology may not be sparse any more [272]. On the other hand, high sampling frequency pushes stable poles of a discrete-time system to the unit circle leading to numerical instability [104]. In this case, it is more reasonable to identifying SDE directly.

9.3.3 Identification of nonlinear models

Black-box linear models are mainly used in this thesis to describe biological networks. Linear models bring several advantages on network inference. First, the construction of linear models requires limited prior knowledge and assumptions on the target network. Inference is performed purely based on experimental data, thus avoiding biased information from subjective assumptions. Second, hidden states can be easily removed from the model, which is particularly useful to cope with unmeasured biological units. Finally, it is more convenient to impose other dynamical properties (e.g. system stability) to the proposed model during inference. Nevertheless, most biological networks contain complex internal dynamics. The relationship among biological units is highly nonlinear. As the dimension of these nonlinear systems grows, linear models cannot provide a good approximation. In addition, linearization of a biological network is valid only if the activities of the system stay in a linear regime, normally, close to the steady state. Often, experiments are not designed to fulfill this condition. Therefore, nonlinear models are more suitable to capture internal dynamics of biological networks.

Since the model structure of nonlinear models is very flexible, the accuracy of the constructed model depends on how much detail of the target network the model can

represent. Constructing a model purely based on the physical background of a biological network requires comprehensive prior knowledge, which is not possible in most applications. While assumptions are normally made based on experimental observations as a compensation, one must take the risk of introducing biased information. Moreover, as the size of the target network grows, making such assumptions becomes prohibitive. To strike a balance between model complexity and accuracy, relaxations are imposed on model structure. For example, nonlinear functions of a model are formulated as a linear combination of basis functions (chapter 4 and 8). This approach is equivalent to constraining the underlying nonlinear functions in a predefined finite dimensional functional space where the functions represent physical laws (e.g. Hill functions and Michaelis-Menten kinetics). In this case, model structure (network topology) is determined by selecting basis functions. The performance of this method replies on how well the system dynamics can be represented by basis functions. To have an accurate approximation, the size of the dictionary may overwhelm the size of data, leading to over-fitting.

To increase the flexibility of model structure for better interpreting system dynamics, black-box nonlinear models can be used, which completely drop physical background and do not specify the working mechanism of biological networks. Kernel methods can be applied to establish an infinite dimensional functional space for nonlinear functions in the model. As a result, the model structure has a much general form under some mild conditions on the property of nonlinear functions such as smoothness and integrability. Identification of this type of model has been widely studied whereas determination of model structure is not a main concern. Under the context of network inference, model structure (network topology) must also be optimized during identification. In the end, model structure is determined by the structure of kernel functions. The ARD technique can be used to optimize the structure of certain kernel functions, for example, radial basis kernel. Nevertheless, this method can perform purely if a kernel function has a complex format, resulting in a highly nonlinear optimization problem. Alternatively, numerical sampling methods such as RJMCMC can be used to encourage a global search of the optimal solution. Another difficulty in applying black-box nonlinear models is hidden nodes. For nonlinear models, hidden nodes cannot be trivially removed in the same way as linear models. Therefore, inference of hidden nodes is inevitable, which brings another problematic issue related to the determination of model complexity.

Bibliography

[1] R. E. Ellis, J. Yuan, and H. R. Horvitz, “Mechanisms and Functions of Cell Death,” *Annu. Rev. Cell Biol.*, vol. 7, no. 1, pp. 663–698, 1991.

[2] H. C. Cheng, C. Ulane, and R. E. Burke, “Clinical progression in Parkinson’s disease and the neurobiology of Axons,” *Ann. Neurol.*, vol. 67, no. 6, pp. 715–725, 2010.

[3] A. Datta, “Genetic engineering for improving quality and productivity of crops,” *Agric. Food Secur.*, vol. 2, no. 1, p. 15, 2013.

[4] L. R. Herrera-estrella, “Genetically Modified Crops and Developing Countries,” vol. 124, no. November, pp. 923–925, 2000.

[5] L. Alberghina and H. V Westerhoff, *Systems Biology: Definitions and Perspectives (Topics in Current Genetics)*. Springer, 2007.

[6] A. E. McKee and P. A. Silver, “Systems biology of gene regulation fulfills its promise,” *Genome Biol.*, vol. 7, no. 5, 2006.

[7] N. Le Novère, “The long journey to a Systems Biology of neuronal function,” *BMC Syst. Biol.*, vol. 1, pp. 7–9, 2007.

[8] J. Strasen, “Cell-specific responses to the cytokine TGFβ are determined by variability in protein levels,” *Mol. Syst. Biol.*, vol. 14, pp. 1–17, 2017.

[9] A. Yachie-Kinoshita, K. Onishi, J. Ostblom, E. Posfai, J. Rossant, and P. W. Zandstra, “Modeling signaling-dependent pluripotent cell states with Boolean logic can predict cell fate transitions,” *bioRxiv*, pp. 1–16, 2017.

[10] S. M. Raimundo, H. M. O. Yang, and E. Massad, “Modeling Vaccine Preventable Vector-borne Infections: Yellow Fever As a Case Study,” *J. Biol. Syst.*, vol. 24, no. 02n03, pp. 193–216, 2016.

[11] P. Hillenbrand, K. C. Maier, P. Cramer, and U. Gerland, “Inference of gene regulation functions from dynamic transcriptome data.” *Elife*, vol. 5, no. September2016, pp. 1-22, 2016

[12] S. M. Hill *et al.*, “Inferring causal molecular networks: empirical assessment through a community-based effort.,” *Nat. Methods*, vol. 13, no. 4, pp. 310–8, 2016.

[13] J. Xiong and T. Zhou, “Structure identification for gene regulatory networks via linearization and robust state estimation,” *Automatica*, vol. 50, no. 11, pp. 2765–2776, 2014.

[14] F. Azuaje, L. Zhang, C. Jeanty, S. L. Puhl, S. Rodius, and D. R. Wagner, “Analysis of a gene co-expression network establishes robust association between Col5a2 and ischemic heart disease,” *BMC Med. Genomics*, vol. 6, no. 1, 2013.

[15] R. C. Meyer *et al.*, “The metabolic signature related to high plant growth rate in *Arabidopsis thaliana*,” *Proc. Natl. Acad. Sci.*, vol. 104, no. 11, pp. 4759–4764, 2007.

[16] J. Lisec *et al.*, “Identification of metabolic and biomass QTL in *Arabidopsis thaliana* in a parallel analysis of RIL and IL populations,” *Plant J.*, vol. 53, no. 6, pp. 960–972, 2008.

[17] N. Hartsfield and G. Ringel, *Pearls in Graph Theory: A Comprehensive Introduction*. Dover Pubn Inc, 2003.

[18] D. J. Wilkinson, *Stochastic Modelling for Systems Biology*. CRC Press, 2012.

[19] E. J. Crampin, S. Schnell, and P. E. McSharry, “Mathematical and computational

techniques to deduce complex biochemical reaction mechanisms," *Prog. Biophys. Mol. Biol.*, vol. 86, no. 1, pp. 77–112, 2004.

[20] J. Ross, "Determination of Complex Reaction Mechanisms. Analysis of Chemical, Biological and Genetic Networks †," *J. Phys. Chem. A*, vol. 112, no. 11, pp. 2134–2143, 2008.

[21] A. Aderhold, D. Husmeier, and M. Grzegorczyk, "Statistical inference of regulatory networks for circadian regulation," *Stat. Appl. Genet. Mol. Biol.*, vol. 13, no. 3, pp. 227–273, 2014.

[22] A. Koryachko, A. Matthiadis, J. J. Ducoste, J. Tuck, T. A. Long, and C. Williams, "Computational approaches to identify regulators of plant stress response using high-throughput gene expression data," *Curr. Plant Biol.*, vol. 3–4, no. July, pp. 20–29, 2015.

[23] W. X. Wang, Y. C. Lai, and C. Grebogi, "Data based identification and prediction of nonlinear and complex dynamical systems," *Phys. Rep.*, vol. 644, no. June, pp. 1–76, 2016.

[24] S. Williams, "Pearson's correlation coefficient.," *N. Z. Med. J.*, vol. 109, no. 1015, p. 38, 1996.

[25] H. L. Kotze *et al.*, "A novel untargeted metabolomics correlation-based network analysis incorporating human metabolic reconstructions," *BMC Syst. Biol.*, vol. 7, p. 107, Oct. 2013.

[26] A. Roy and C. B. Post, "Detection of Long-Range Concerted Motions in Protein by a Distance Covariance," *J. Chem. Theory Comput.*, vol. 8, no. 9, pp. 3009–3014, Sep. 2012.

[27] J. Kong, S. Wang, and G. Wahba, "Using distance covariance for improved variable selection with application to learning genetic risk models," *Stat. Med.*, vol. 34, no. 10, pp. 1708–1720, 2015.

[28] S. Kumari *et al.*, "Evaluation of Gene Association Methods for Coexpression Network Construction and Biological Knowledge Discovery," *PLoS One*, vol. 7, no. 11, 2012.

[29] H. Peng, S. Wang, and X. Wang, "Consistency and asymptotic distribution of the Theil-Sen estimator," *J. Stat. Plan. Inference*, vol. 138, no. 6, pp. 1836–1850, 2008.

[30] A. Reverter and E. K. F. Chan, "Combining partial correlation and an information theory approach to the reversed engineering of gene co-expression networks," *Bioinformatics*, vol. 24, no. 21, pp. 2491–2497, 2008.

[31] A. de la Fuente, N. Bing, I. Hoeschele, and P. Mendes, "Discovery of meaningful associations in genomic data using partial correlation coefficients," *Bioinformatics*, vol. 20, no. 18, pp. 3565–3574, 2004.

[32] M. G. Kendall, "A New Measure of Rank Correlation," *Biometrika,* vol. 30, no. 1, pp. 81–93, 1938.

[33] S. Hempel, A. Koseska, J. Kurths, and Z. Nikoloski, "Inner composition alignment for inferring directed networks from short time series," *Phys. Rev. Lett.*, vol. 107, no. 5, pp. 2–5, 2011.

[34] A. Fujita, J. R. Sato, M. A. A. Demasi, M. C. Sogayar, C. E. Ferreira, and S. Miyano, "Comparing Pearson, Spearman and Hoeffding D Measure for Gene Expression Association Analysis," *J. Bioinform. Comput. Biol.*, vol. 7, no. 4, pp. 663–684, 2009.

[35] S. L. Bressler and A. K. Seth, "Wiener-Granger Causality: A well established methodology," *Neuroimage*, vol. 58, no. 2, pp. 323–329, 2011.

[36] A. Porta and L. Faes, “Wiener-Granger Causality in Network Physiology with Applications to Cardiovascular Control and Neuroscience,” *Proc. IEEE*, vol. 104, no. 2, pp. 282–309, 2016.

[37] H. Ye, E. R. Deyle, L. J. Gilarranz, and G. Sugihara, “Distinguishing time-delayed causal interactions using convergent cross mapping,” *Sci. Rep.*, vol. 5, pp. 1–9, 2015.

[38] A. T. Clark *et al.*, “Spatial convergent cross mapping to detect causal relationships from short time series,” *Ecology*, vol. 96, no. 5, pp. 1174–1181, 2015.

[39] D. Mønster, R. Fusaroli, K. Tylén, A. Roepstorff, and J. F. Sherson, “Causal inference from noisy time-series data — Testing the Convergent Cross-Mapping algorithm in the presence of noise and external influence,” *Futur. Gener. Comput. Syst.*, vol. 73, pp. 52–62, 2017.

[40] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[41] J. B. Kinney and G. S. Atwal, “Equitability, mutual information, and the maximal information coefficient,” *Proc. Natl. Acad. Sci.*, vol. 111, no. 9, pp. 3354–3359, 2014.

[42] M. Moriyama *et al.*, “Relevance Network between Chemosensitivity and Transcriptome in Human Hepatoma Cells1,” *Mol. Cancer Ther.*, vol. 2, no. 2, pp. 199–205, 2003.

[43] A. J. Butte, P. Tamayo, D. Slonim, T. R. Golub, and I. S. Kohane, “Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks,” *Proc. Natl. Acad. Sci.*, vol. 97, no. 22, pp. 12182–12186, 2000.

[44] A. Madar, A. Greenfield, E. Vanden-Eijnden, and R. Bonneau, “DREAM3: Network inference using dynamic context likelihood of relatedness and the inferelator,” *PLoS One*, vol. 5, no. 3, 2010.

[45] A. Unler, A. Murat, and R. B. Chinnam, “mr2PSO: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification,” *Inf. Sci. (Ny).*, vol. 181, no. 20, pp. 4625–4641, 2011.

[46] A. A. Margolin *et al.*, “ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context,” *BMC Bioinformatics*, vol. 7, no. SUPPL.1, pp. 1–15, 2006.

[47] X. Zhao, W. Deng, and Y. Shi, “Feature selection with attributes clustering by maximal information coefficient,” *Procedia Comput. Sci.*, vol. 17, pp. 70–79, 2013.

[48] D. Reshef *et al.*, “Detecting Novel Associations in Large Data Sets,” *Sci. Transl. Med.*, vol. 334, no. 6062, pp. 1518–1524, 2011.

[49] S. Ma, Q. Gong, and H. J. Bohnert, “An *Arabidopsis* gene network based on the graphical Gaussian model,” *Genome Res.*, vol. 17, no. 11, pp. 1614–1625, Nov. 2007.

[50] X. Wu, Y. Ye, and K. R. Subramanian, “Interactive Analysis of Gene Interactions Using Graphical Gaussian Model,” in *Proceedings of the 3rd International Conference on Data Mining in Bioinformatics*, 2003, pp. 63–69.

[51] J. Friedman, T. Hastie, and R. Tibshirani, “Sparse inverse covariance estimation with the graphical lasso,” *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[52] N. Friedman, D. Geiger, and M. Goldszmit, “Bayesian Network Classifiers,” *Mach. Learn.*, vol. 29, pp. 131–163, 1997.

[53] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis, “Advances to Bayesian network inference for generating causal networks from observational biological data,” *Bioinformatics*, vol. 20, no. 18, pp. 3594–3603, 2004.

[54] P. Dagum and M. Luby, “Approximating probabilistic inference in Bayesian belief networks is NP-hard,” *Artif. Intell.*, vol. 60, no. 1, pp. 141–153, 1993.

[55] P. Dagum and M. Luby, “An optimal approximation algorithm for Bayesian inference,” *Artif. Intell.*, vol. 93, no. 1–2, pp. 1–27, 1997.

[56] I. Tsamardinos, C. F. Aliferis, and A. Statnikov, “Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations,” in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 673–678.

[57] C. F. Aliferis Constantinaliferis, A. Statnikov Alexanderstatnikov, S. Mani, and X. D. Koutsoukos, “Local Causal and Markov Blanket Induction for Causal Discovery and Feature Selection for Classification Part I: Algorithms and Empirical Evaluation,” *J. Mach. Learn. Res.*, vol. 11, pp. 171–234, 2010.

[58] N. Dalchau, “Mathematical Modelling of Circadian Signalling in *Arabidopsis*,” University of Cambridge, 2008.

[59] J. C. W. Locke, A. J. Millar, and M. S. Turner, “Modelling genetic networks with noisy and varied experimental data: The circadian clock in *Arabidopsis thaliana*,” *J. Theor. Biol.*, vol. 234, pp. 383–393, 2005.

[60] J. C. W. Locke *et al.*, “Experimental validation of a predicted feedback loop in the multi-oscillator clock of *Arabidopsis thaliana*.,” *Mol. Syst. Biol.*, vol. 2, p. 59, 2006.

[61] A. Pokhilko *et al.*, “Data assimilation constrains new connections and components in a complex, eukaryotic circadian clock model.,” *Mol. Syst. Biol.*, vol. 6, no. 416, pp. 1–10, 2010.

[62] A. Pokhilko, A. P. Fernández, K. D. Edwards, M. M. Southern, K. J. Halliday, and A. J. Millar, “The clock gene circuit in *Arabidopsis* includes a repressilator with additional feedback loops,” *Mol. Syst. Biol.*, no. 574, pp. 1–13, 2012.

[63] A. Pokhilko, P. Mas, and A. J. Millar, “Modelling the widespread effects of TOC1 signalling on the plant circadian clock and its outputs.,” *BMC Syst. Biol.*, vol. 7, no. 1, p. 23, 2013.

[64] T. Ohara, H. Fukuda, and I. T. Tokuda, “An extended mathematical model for reproducing the phase response of *Arabidopsis thaliana* under various light conditions,” *J. Theor. Biol.*, vol. 382, pp. 337–344, 2015.

[65] W. Li, J. Feng, and T. Jiang, “IsoLasso: A LASSO regression approach to RNA-Seq based transcriptome assembly (Extended abstract),” *J. Comput. Biol.*, vol. 18, no. 11, pp. 168–188, 2011.

[66] R. J. Tibshirani, “The lasso method for variable selection in the Cox model,” *Stat. Med.*, vol. 16, no. March 1995, pp. 385–395, 1997.

[67] M. Gustafsson, M. Hörnquist, and A. Lombardi, “Constructing and analyzing a large-scale gene-to-gene regulatory network-lasso-constrained inference and biological validation,” *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 2, no. 3, pp. 254–261, 2005.

[68] W. Pan, Y. Yuan, J. Gonçalves, and G. Stan, “A Sparse Bayesian Approach to the Identification of Nonlinear State-Space Systems,” *IEEE Trans. Autom. Control*, vol. 61, pp. 182–187, 2016.

[69] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *J.R. Stat. Soc. B*, vol. 67, no. 2, pp. 301–320, 2005.

[70] A. Ahmed and E. P. Xing, “Recovering time-varying networks of dependencies in social

and biological studies," *PNAS*, vol. 106, no. 29, pp. 1–6, 2009.

[71] M. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.

[72] M. Grzegorczyk, "A Non-Homogeneous Dynamic Bayesian Network with Sequentially Coupled Interaction Parameters for Applications in Systems and Synthetic Biology," *Stat. Appl. Genet. Mol. Biol.*, vol. 11, no. 4, 2012.

[73] E. R. Morrissey, M. A. Juárez, K. J. Denby, and N. J. Burroughs, "Inferring the time-invariant topology of a nonlinear sparse gene regulatory network using fully Bayesian spline autoregression," *Biostatistics*, vol. 12, no. 4, pp. 682–694, 2011.

[74] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning.*, Illustrate. MIT Press, 2006.

[75] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian Process Dynamical Models for Human Motion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 283–298, 2008.

[76] A. Aderhold, D. Husmeier, and M. Grzegorczyk, "Approximate Bayesian inference in semi-mechanistic models," *Stat. Comput.*, no. May, pp. 1–38, 2016.

[77] S. Huang *et al.*, "A Sparse Structure Learning Algorithm for Gaussian Bayesian Network Identification from High-Dimensional Data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1328–1342, 2013.

[78] D. Geiger and D. Heckerman, "Learning Gaussian Networks," in *International Conference on Uncertainty in Artificial Intelligence*, 1994, pp. 235–243.

[79] Y. Ko, C. Zhai, and S. L. Rodriguez-Zas, "Inference of Gene Pathways Using Gaussian Mixture Models," in *2007 IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2007)*, 2007, pp. 362–367.

[80] Y. Ko, C. Zhai, and S. Rodriguez-Zas, "Inference of gene pathways using mixture Bayesian networks.," *BMC Syst. Biol.*, vol. 3, p. 54, 2009.

[81] K. P. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning," *Ann. Phys. (N. Y).*, vol. Ph. D., p. 225, 2002.

[82] Z. Ghahramani, "Learning Dynamic Bayesian Networks," *Adapt. Process. Seq. Data Struct.*, vol. 1387, pp. 168–197, 1997.

[83] C. A. Pollino and C. Henderson, "Bayesian networks : A guide for their application in natural resource management and policy," 2010.

[84] J. Goncalves and S. Warnick, "Necessary and sufficient conditions for dynamical structure reconstruction of LTI networks," *IEEE Trans. Automat. Contr.*, vol. 53, no. 7, pp. 1670–1674, 2008.

[85] Y. Yuan, K. Glover, and J. Gonçalves, "On minimal realisations of dynamical structure functions," *Automatica*, vol. 55, no. December, pp. 159–164, 2015.

[86] K. P. Burnham and D. R. Anderson, *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach (2nd ed)*, vol. 172. 2002.

[87] P. P. J. van den Bosch and A. C. van der Klauw, *Modeling, Identification and Simulation of Dynamical Systems*. CRC Press, 1994.

[88] P. M. J. Van Den Hof, *System Identification for Control*, 2004.

[89] L. Ljung, *System Identification: Theory for the User*. Prentice Hall, 1998.

[90] M. Jordan, J. Kleinberg, and B. Scho, *Pattern Recognition and Machine Learning*.

Springer New York, 2006.

[91] T. Chen, M. S. Andersen, L. Ljung, A. Chiuso, and G. Pillonetto, “System identification via sparse multiple kernel-based regularization using sequential convex optimization techniques,” *IEEE Trans. Automat. Contr.*, vol. 59, no. 11, pp. 1–33, 2014.

[92] Z. M. Kassas, “Numerical simulation of continuous-time stochastic dynamical systems with noisy measurements and their discrete-time equivalents,” *Proc. IEEE Int. Symp. Comput. Control Syst. Des.*, pp. 1397–1402, 2011.

[93] D. J. Higham., “An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations,” *SIAM Rev.*, vol. 43, no. 3, pp. 525–546, 2001.

[94] G. P. Rao, G. P. Rao, H. Garnier, and H. Garnier, “Numerical illustrations of the relevance of direct continuous-time model identification,” *15th Trienn. IFAC World Congr. Autom. Control*, 2002.

[95] D. Gillespie, “Exact stochastic simulation of coupled chemical reactions,” *J. Phys. Chem.*, vol. 81, no. 25, pp. 2340–2361, 1977.

[96] D. T. Gillespie, “The chemical Langevin equation,” *J. Chem. Phys.*, vol. 113, no. 1, pp. 297–306, 2000.

[97] D. T. Gillespie, “The multivariate Langevin and Fokker–Planck equations,” *Am. J. Phys.*, vol. 64, no. 10, p. 1246, 1996.

[98] J. D. Murray, *Mathematical Biology: I . An Introduction, Third Edition*, vol. 1, no. 1. 2002.

[99] E. H. Flach and S. Schnell, “Use and abuse of the quasi-steady-state approximation,” *IEE Proc. - Syst. Biol.*, vol. 153, no. 4, p. 187, 2006.

[100] J. C. W. Locke *et al.*, “Extension of a genetic network model by iterative experimentation and mathematical analysis.,” *Mol. Syst. Biol.*, vol. 1, p. 2005.0013, 2005.

[101] N. Dalchau *et al.*, “Correct biological timing in *Arabidopsis* requires multiple light-signaling pathways.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 107, pp. 13171–13176, 2010.

[102] A. Carignano, J. Junyang, A. Webb, and J. Gonçalves, “Assessing the effect of unknown widespread perturbations in complex systems using the v-gap,” in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 3193–3198.

[103] A. Carignano, “Genome wide analysis of differentially expressed systems: an application to circadian networks,” 2014.

[104] H. Garnier and L. Wang, *Identification of Continous time Models from Sampled Data*. Springer; 2008 edition, 2008.

[105] J. Kocijan, *Modelling and control of dynamic systems using gaussian process models*. Springer; 1st ed. 2016 edition, 2015.

[106] R. S. Risuleo, “System identification with input uncertainties: an EM kernel-based approach,” 2016.

[107] D. P. Wipf, “Bayesian Methods for Finding Sparse Representations,” 2006.

[108] D. P. Wipf, B. D. Rao, and S. Nagarajan, “Latent Variable Bayesian Models for Promoting Sparsity,” *IEEE Trans. Inf. Theory*, vol. 57, no. 9, pp. 6236–6255, 2011.

[109] W. Pan, A. Sootla, and G. B. Stan, “Distributed reconstruction of nonlinear networks: An ADMM approach,” *IFAC Proc. Vol.*, vol. 19, pp. 3208–3213, 2014.

[110] W. Pan, Y. Yuan, L. Ljung, J. Gonçalves, and G. Stan, “Identifying biochemical reaction networks from heterogeneous datasets,” in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 2525–2530.

[111] W. Pan, Y. Yuan, H. Sandberg, J. Gonçalves, and G.B. Stan, “Online fault diagnosis for nonlinear power systems,” *Automatica*, vol. 55, no. MAY, pp. 27–36, 2015.

[112] T. Chen, H. Ohlsson, and L. Ljung, “On the estimation of transfer functions, regularizations and Gaussian processes- Revisited,” *Automatica*, vol. 48, no. 48, pp. 1525–1535, 2012.

[113] A. Chiuso and G. Pillonetto, “A Bayesian approach to sparse dynamic network identification,” *Automatica*, vol. 48, no. 8, pp. 1553–1565, 2012.

[114] M. Darwish, P. Cox, G. Pillonetto, and R. Tóth, “Bayesian Identification of LPV Box-Jenkins Models,” *Proc. 54th IEEE Conf. Decis. Control (CDC), Osaka, Japan*, no. Cdc, 2015.

[115] R. A. Kennedy and P. Sadeghi, *Hilbert Space Methods in Signal Processing*. Cambridge University Press, 2013.

[116] G. Pillonetto, T. Chen, A. Chiuso, G. De Nicolao, and L. Ljung, “Regularized linear system identification using atomic, nuclear and kernel-based norms: The role of the stability constraint,” *Automatica*, vol. 69, pp. 137–149, 2016.

[117] G. Pillonetto and A. Chiuso, “Tuning complexity in regularized kernel-based regression and linear system identification: The robustness of the marginal likelihood estimator,” *Automatica*, vol. 58, pp. 106–117, 2015.

[118] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, “Kernel methods in system identification, machine learning and function estimation: A survey,” *Automatica*, vol. 50, no. 3, pp. 657–682, 2014.

[119] G. Wahba, *Spline Models for Observational Data*. SIAM: Society for Industrial and Applied Mathematics, 1990.

[120] P. L. Green, K. Worden, M. Street, and S. Sheffield, “Bayesian and Markov chain Monte Carlo methods for identifying nonlinear systems in the presence of uncertainty,” *Philos. Trans. R. Soc. A*, vol. 373, 2015.

[121] D. Tiboaca, P. L. Green, R. J. Barthorpe, and K. Worden, “Bayesian System Identification of Dynamical Systems Using Reversible Jump Markov Chain Monte Carlo,” in *Topics in Modal Analysis II, Volume 8*, 2014, pp. 277–284.

[122] P. L. Green, “Bayesian system identification of a nonlinear dynamica system using a novel variant of Simulated Annealing,” *Mech. Syst. Signal Process.*, vol. 52–53, no. 1, pp. 133–146, 2015.

[123] P. L. Green, “Bayesian system identification of dynamical systems using large sets of training data: A MCMC solution,” *Probabilistic Eng. Mech.*, vol. 42, pp. 54–63, 2015.

[124] S. Han, R. K. W. Wong, T. C. M. Lee, L. Shen, S. Y. R. Li, and X. Fan, “A Full Bayesian Approach for Boolean Genetic Network Inference,” *PLoS One*, vol. 9, no. 12, p. e115806, Dec. 2015.

[125] N. B. Agostinho, K. S. Machado, and A. V Werhli, “Inference of regulatory networks with a convergence improved MCMC sampler,” *BMC Bioinformatics*, pp. 1–10, 2015.

[126] P. J. Green, “Reversible jump Markov chain monte carlo computation and Bayesian model determination,” *Biometrika*, vol. 82, no. 4, pp. 711–732, 1995.

[127] R. J. Polland, *Essentials of Robust Control*. 2005.

[128] L. Mombaerts, A. Mauroy, and J. Gonçalves, “Optimising time-series experimental design for modelling of circadian rhythms: the value of transient data,” *IFAC-PapersOnLine*, vol. 49, no. 26, pp. 109–113, 2016.

[129] L. Mombaerts *et al.*, "Dynamical Differential Expression (DyDE) Reveals the Period Control Mechanisms of the *Arabidopsis* Circadian Oscillator (submitted)," *PLoS Comput. Biol.*, 2018.

[130] D. Marbach, R. J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, and G. Stolovitzky, "Revealing strengths and weaknesses of methods for gene network inference," *Proc. Natl. Acad. Sci.*, vol. 107, no. 14, pp. 6286–6291, 2010.

[131] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, p. 651, Oct. 2000.

[132] H. Jeong, S. P. Mason, A. L. Barabási, and Z. N. Oltvai, "Lethality and centrality in protein networks," *Nature*, vol. 411, no. 6833, pp. 41–42, 2001.

[133] A. Wagner and D. A. Fell, "The small world inside large metabolic networks," *Proc. R. Soc. B Biol. Sci.*, vol. 268, no. 1478, pp. 1803–1810, 2001.

[134] P. Qiu, A. J. Gentles, and S. K. Plevritis, "Fast calculation of pairwise mutual information for gene regulatory network reconstruction," *Comput. Methods Programs Biomed.*, vol. 94, no. 2, pp. 177–180, 2009.

[135] A. J. Butte and I. S. Kohane, "Mutual Information Relevance Networks: Functional Genomic Clustering Using Pairwise Entropy Measurements," in *Biocomputing 2000*, 1999, pp. 418–429.

[136] A. C. Damianou, M. K. Titsias, and N. D. Lawrence, "Variational Inference for Latent Variables and Uncertain Inputs in Gaussian Processes," *J. Mach. Learn. Res.*, vol. 17, pp. 1–62, 2016.

[137] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian Processes for Big Data," in *UAI*, 2013, pp. 282–290.

[138] R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso," *J. R. Stat. Soc. Ser. B*, vol. 58, pp. 267–288, 1994.

[139] P. Congdon, "Applied Bayesian hierarchical methods," *J. Appl. Stat.*, vol. 39, no. 8, pp. 1845–1845, 2010.

[140] S. Gibson and B. Ninness, "Robust maximum-likelihood estimation of multivariable dynamic systems," *Automatica*, vol. 41, pp. 1667–1682, 2005.

[141] a. Wills, B. Ninness, and S. Gibson, "Maximum Likelihood Estimation of State Space Models from Frequency Domain Data," *IEEE Trans. Automat. Contr.*, vol. 54, no. 1, 2009.

[142] M. Schmidt, "Least Squares Optimization with L1-Norm Regularization," *Optimization*, vol. 98, no. December, pp. 230–238, 2005.

[143] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A Sparse-Group Lasso," *J. Comput. Graph. Stat.*, vol. 22, no. 2, pp. 231–245, 2013.

[144] E. J. Candès, M. B. Wakin, and S. P. Boyd, "Enhancing Sparsity by Reweighted ℓ 1 Minimization," *J. Fourier Anal. Appl.*, vol. 14, no. 5–6, pp. 877–905, 2008.

[145] D. Hayden, Y. H. Chang, J. Goncalves, and C. J. Tomlin, "Sparse network identifiability via Compressed Sensing," *Automatica*, vol. 68, pp. 9–17, 2016.

[146] Y. C. Eldar, P. Kuppinger, and H. Bölcskei, "Block-sparse signals: Uncertainty relations and efficient recovery," *IEEE Trans. Signal Process.*, vol. 58, no. 6, pp. 3042–3054, 2010.

[147] B. M. Sanandaji, T. L. Vincent, and M. B. Wakin, "Exact topology identification of large-scale interconnected dynamical systems from compressive observations," in *Proceedings of the 2011 American Control Conference*, 2011, pp. 649–656.

[148] S. D. Babacan, S. Nakajima, and M. N. Do, “Bayesian group-sparse modeling and variational inference,” *IEEE Trans. Signal Process.*, vol. 62, no. 11, pp. 2906–2921, 2014.

[149] R. B. Chen, C. H. Chu, S. Yuan, and Y. N. Wu, “Bayesian Sparse Group Selection,” *J. Comput. Graph. Stat.*, 2015.

[150] S. Boyd and L. Vandenberghe, *Convex Optimization*, vol. 25, no. 3. 2010.

[151] K. Kreutz-delgado and B. D. Rao, “A General Approach to Sparse Basis Selection: Majorization, Concavity, and Affine Scaling,” *Comput. Eng.*, p. 49, 1997.

[152] Y. C. Eldar and G. Kutyniok, *Compressed Sensing*. Cambridge University Press, 2012.

[153] J. Dattorro, *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing, 2015.

[154] A. Aravkin, J. V. Burke, A. Chiuso, and G. Pillonetto, “Convex vs nonconvex approaches for sparse estimation: Lasso, Multiple Kernel Learning and Hyperparameter Lasso,” in *Proceedings of the IEEE Conference on Decision and Control*, 2011, pp. 156–161.

[155] Z. Qin, K. Scheinberg, and D. Goldfarb, “Efficient block-coordinate descent algorithms for the Group Lasso,” *Math. Program. Comput.*, vol. 5, no. 2, pp. 143–169, 2013.

[156] M. Yuan and Y. Lin, “Model selection and estimation in regression with grouped variables,” *J. R. Stat. Soc. Ser. B (Statistical Methodol.)*, vol. 68, no. 1, pp. 49–67.

[157] M. Vincent and N. R. Hansen, “Sparse group lasso and high dimensional multinomial classification,” *Comput. Stat. Data Anal.*, vol. 71, pp. 771–786, 2014.

[158] J. B. Bell, A. N. Tikhonov, and V. Y. Arsenin, “Solutions of Ill-Posed Problems.,” *Math. Comput.*, vol. 32, no. 144, p. 1320, 1978.

[159] S. Society and S. B. Methodological, “Regression Shrinkage and Selection via the Lasso Robert Tibshirani,” *J.R. Stat. Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.

[160] J. D. Hamilton, *Time Series Analysis*, 1994.

[161] D. Wipf and S. Nagarajan, “A New View of Automatic Relevance Determination,” *Compute*, vol. 20, no. 2, pp. 1625–1632, 2008.

[162] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*. Springer-Verlag, 1993.

[163] R. Giri and B. D. Rao, “Type I and Type II Bayesian Methods for Sparse Signal Recovery using Scale Mixtures,” *IEEE Trans. Signal Process.*, vol. 64, no. 13, pp. 3418–3428, 2016.

[164] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, “An introduction to MCMC for machine learning,” *Mach. Learn.*, vol. 50, no. 1–2, pp. 5–43, 2003.

[165] C. Geyer, “Markov chain Monte Carlo lecture notes,” 1998.

[166] J. Palmer, D.P. Wipf, K. Kreutz-Delgado, and B.D. Rao, “Variational EM algorithms for non-Gaussian latent variable models,” *Adv. Neural Inf. Process. Syst.*, vol. 18, pp. 1059–1066, 2006.

[167] D. P. Wipf and B. D. Rao, “Comparing the Effects of Different Weight Distributions on Finding Sparse Representations,” *Adv. Neural Inf. Process. Syst. 18*, no. 1, pp. 1521–1528, 2006.

[168] M. A. T. Figueiredo, “Adaptive Sparseness using Jeffreys’ Prior,” *Adv. Neural Inf. Process. Syst.*, vol. 1, pp. 697–704, 2002.

[169] Z. Zhang and B. D. Rao, “Extension of SBL algorithms for the recovery of block sparse signals with intra-block correlation,” *IEEE Trans. Signal Process.*, vol. 61, no. 8, pp. 2009–2015, 2013.

[170] D. P. Wipf and B. D. Rao, “An empirical Bayesian strategy for solving the simultaneous sparse approximation problem,” *IEEE Trans. Signal Process.*, vol. 55, no. 7 II, pp. 3704–3716, 2007.

[171] K. Zhou, J. C. Doyle, and K. Glover, *Robust and optimal control*. 1996.

[172] Z. Yue, J. Thunberg, W. Pan, L. Ljung, and J. Goncalves, “Linear Dynamic Network Reconstruction from Heterogeneous Datasets,” 2016.

[173] J. S. Sakellariou and S. D. Fassois, “Nonlinear ARX (NARX) based identification and fault detection in a 2 DOF system with cubic stiffness,” in *International Conference on noise and vibration engineering*, 2002.

[174] C. Andrieu, N. de Freitas, and A. Doucet, “Robust full Bayesian learning for radial basis networks.,” *Neural Comput.*, vol. 13, no. 10, pp. 2359–2407, 2001.

[175] D. Wipf and S. Nagarajan, “Iterative Reweighted l1 and l2 Methods for Finding Sparse Solutions,” *IEEE J. Sel. Top. Signal …*, vol. 4, no. 2, pp. 317–329, 2010.

[176] B. K. Sriperumbudur, “On the Convergence of the Concave-Convex Procedure,” *Nips*, no. 1, pp. 1–9, 2009.

[177] S. Boyd, “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,” *Found. Trends® Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.

[178] H. A. Le Thi, T. Pham Dinh, H. M. Le, and X. T. Vo, “DC approximation approaches for sparse optimization,” *Eur. J. Oper. Res.*, vol. 244, no. 1, pp. 26–46, 2015.

[179] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. Wiley-Interscience; 2 edition, 2008.

[180] T. Baldacchino, S. R. Anderson, and V. Kadirkamanathan, “Computational system identification for Bayesian NARMAX modelling,” *Automatica*, vol. 49, no. 9, pp. 2641–2651, 2013.

[181] H. Sun, “Mercer theorem for RKHS on noncompact sets,” *J. Complex.*, vol. 21, no. 3, pp. 337–349, 2005.

[182] G. Pillonetto, M. H. Quang, and A. Chiuso, “A new kernel-based approach for nonlinearsystem identification,” *IEEE Trans. Automat. Contr.*, vol. 56, no. 12, pp. 2825–2840, 2011.

[183] G. Pillonetto and G. De Nicolao, “A new kernel-based approach for linear system identification,” *Automatica*, vol. 46, no. 1, pp. 81–93, 2010.

[184] G. Bottegal, A. Y. Aravkin, H. Hjalmarsson, and G. Pillonetto, “Robust EM kernel-based methods for linear system identification,” *Automatica*, vol. 67, pp. 114–126, 2016.

[185] Z. Gillani, M. S. A. Akash, M. M. Rahaman, and M. Chen, “CompareSVM: Supervised, Support Vector Machine (SVM) inference of gene regularity networks,” *BMC Bioinformatics*, vol. 15, no. 1, pp. 1–7, 2014.

[186] J. P. Vert and Y. Yamanishi, “Supervised graph inference,” *Adv. Neural Inf. Process. Syst.*, vol. 17, pp. 1433–1440, 2005.

[187] Y. Yamanishi, J. P. Vert, and M. Kanehisa, “Protein network inference from multiple genomic data: A supervised approach,” *Bioinformatics*, vol. 20, no. SUPPL. 1, pp. 363–370, 2004.

[188] M. Kotera, Y. Yamanishi, Y. Moriya, M. Kanehisa, and S. Goto, “GENIES: Gene network inference engine based on supervised analysis,” *Nucleic Acids Res.*, vol. 40, no. W1, pp. 162–167, 2012.

[189] J. P. Vert, J. Qiu, and W. S. Noble, “A new pairwise kernel for biological network inference with support vector machines,” *BMC Bioinformatics*, vol. 8, no. SUPPL. 10, pp. 1–10, 2007.

[190] T. Kato, K. Tsuda, and K. Asai, “Selective integration of multiple biological data for supervised network inference,” *Bioinformatics*, vol. 21, no. 10, pp. 2488–2495, 2005.

[191] W. Rudin, *Functional Analysis*. McGraw-Hill Science/Engineering/Math; 2 edition, 1991.

[192] P. D. Lax, *Functional analysis*. Wiley-Interscience; 1 edition, 2002.

[193] D. Sejdinovic and A. Gretton, “What is an RKHS?,” 2012.

[194] M. H. Quang, “Reproducing Kernel Hilbert Spaces in Learning Theory,” Brown University, 2006.

[195] F. Cucker and S. Smale, “On the mathematical foundations of learning,” *Bull. Am. Math. Soc.*, vol. 39, no. 1, pp. 1–49, 2002.

[196] N. Aronszajn, “Theory of reproducing kernels,” *Trans. Am. Math. Soc.*, vol. 68, no. 3, pp. 337–337, 1950.

[197] S. on T. S. Analysis and M. Rosenblatt, “Proceedings of a symposium on time series analysis.,” 1963.

[198] G. Pillonetto, A. Chiuso, and G. De Nicolao, “Prediction error identification of linear systems: A nonparametric Gaussian regression approach,” *Automatica*, vol. 47, no. 2, pp. 291–305, 2011.

[199] F. Dinuzzo, “Kernels for linear time invariant system identification,” *SIAM J. Control OPTIM.*, vol. 53, no. 5, pp. 1–17, 2015.

[200] Y. Yuan, G. B. Stan, S. Warnick, and J. Goncalves, “Robust dynamical network structure reconstruction,” *Automatica*, vol. 47, no. 6, pp. 1230–1235, 2011.

[201] F. P. Carli, T. Chen, and L. Ljung, “Maximum Entropy Kernels for System Identification,” *IEEE Trans. Automat. Contr.*, vol. 62, no. 3, pp. 1471–1477, 2017.

[202] T. Chen, T. Ardeshiri, F. P. Carli, A. Chiuso, L. Ljung, and G. Pillonetto, “Maximum entropy properties of discrete-time first-order stable spline kernel,” *Automatica*, vol. 66, pp. 34–38, 2016.

[203] R. Waagepetersen and D. Sorensen, “A Tutorial on Reversible Jump MCMC with a View toward Applications in QTL-mapping,” *Int. Stat. Rev.*, vol. 69, no. 1, pp. 49–61, May 2007.

[204] H. Nastase, “Introduction to Markov Chain Monte Carlo,” in *Handbook of Markov Chain Monte Carlo*, 2007, pp. 3–48.

[205] D. I. Hastie, “Towards automatic reversible jump Markov chain Monte Carlo,” 2005.

[206] C. Andrieu and J. Thoms, “A tutorial on adaptive MCMC,” *Stat. Comput.*, vol. 18, no. 4, pp. 343–373, 2008.

[207] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, “Introducing Markov Chain Monte Carlo,” *Markov Chain Monte Carlo in Practice*. p. 512, 1996.

[208] G. O. Roberts and S. K. Sahu, “Updating Schemes, Correlation Structure, Blocking and Parameterization for the Gibbs Sampler,” *J. R. Stat. Soc. Ser. B*, vol. 59, no. 2, pp. 291–317, 1997.

[209] A. A. Johnson, G. L. Jones, and R. C. Neath, “Component-Wise Markov Chain Monte Carlo: Uniform and Geometric Ergodicity under Mixing and Composition,” *Stat. Sci.*, vol. 28, no. 3, pp. 360–375, 2013.

[210] D. A. van Dyk and X. Jiao, “Metropolis-Hastings Within Partially Collapsed Gibbs

Samplers," *J. Comput. Graph. Stat.*, vol. 24, no. 2, pp. 301–327, 2015.

[211] D. A. van Dyk and T. Park, "Partially Collapsed Gibbs Samplers: Theory and Methods," *J. Am. Stat. Assoc.*, vol. 103, no. 482, pp. 790–796, 2008.

[212] J. S. Liu, "The Collapsed Gibbs Sampler in Bayesian Computations with Applications to a Gene Regulation Problem," *J. Am. Stat. Assoc.*, vol. 89, no. 427, pp. 958–966, 1994.

[213] P. J. Green and D. I. Hastie, "Reversible jump MCMC," *Genetics*, vol. 155, no. 3, pp. 1391–1403, 2009.

[214] S. P. Brooks and B. J. T. Morgan, "Optimization Using Simulated Annealing," *J. R. Stat. Soc.*, vol. 44, no. 2, pp. 241–257, 1995.

[215] S. P. Brooks, N. Friel, and R. King, "Classical Model Selection via Simulated Annealing," *J. R. Stat. Soc. B*, vol. 65, no. 2, pp. 503–520, 2003.

[216] C. Andrieu, N. De Freitas, and A. Doucet, "Reversible jump MCMC simulated annealing for neural networks," *Uncertaintity in Artificial Intelligence Proceedings*, pp. 11–18, 2000.

[217] P. T. Troughton and S. J. Godsill, "A reversible jump sampler for autoregressive time series," *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98.*, vol. 4, pp. 2257–2260, 1998.

[218] C. Andrieu and A. Doucet, "Joint Bayesian Model Selection and Estimation of Noisy Sinusoids via Reversible Jump MCMC,'" *IEEE Trans. Signal Process.*, vol. 47, no. 10, pp. 2667–2676, 1999.

[219] J. Vermaak, C. Andrieu, A. Doucet, and S. J. Godsill, "Reversible jump Markov chain Monte Carlo strategies for Bayesian model selection in autoregressive processes," *J. Time Ser. Anal.*, vol. 25, no. 6, pp. 785–809, 2004.

[220] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. 2004.

[221] S. Ambikasaran, D. Foreman-mackey, L. Greengard, D. W. Hogg, and M. O. Neil, "Fast Direct Methods for Gaussian Processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 252–265, 2016.

[222] S. Ambikasaran and E. Darve, "An O(N log N) Fast Direct Solver for Partial Hierarchically Semi-Separable Matrices: With Application to Radial Basis Function Interpolation," *J. Sci. Comput.*, vol. 57, no. 3, pp. 477–501, 2013.

[223] P. Y. Hsu and S. L. Harmer, "Wheels within wheels: The plant circadian system," *Trends Plant Sci.*, vol. 19, no. 4, pp. 240–249, 2014.

[224] S. Chib and I. Jeliazkov, "Marginal Likelihood from The Metropolis – Hastings Output," *J. Am. Stat. Assoc.*, vol. 96, no. 453, pp. 270–81, 2001.

[225] A. N. Dodd *et al.*, "The *Arabidopsis* circadian clock incorporates a cADPR-based feedback loop," *Science*, vol. 318, no. 5857, pp. 1789–1792, 2007.

[226] M. J. Haydon, O. Mielczarek, F. C. Robertson, K. E. Hubbard, and A. A. R. Webb, "Photosynthetic entrainment of the *Arabidopsis thaliana* circadian clock," *Nature*, vol. 502, no. 7473, pp. 689–692, 2013.

[227] U. Voß *et al.*, "The circadian clock rephases during lateral root organ initiation in *Arabidopsis thaliana*," *Nat. Commun.*, vol. 6, 2015.

[228] S. L. Harmer *et al.*, "Orchestrated transcription of key pathways in *Arabidopsis* by the circadian clock," *Science*, vol. 290, no. 5499, pp. 2110–2113, 2000.

[229] E. Herrero *et al.*, "EARLY FLOWERING4 recruitment of EARLY FLOWERING3 in the

nucleus sustains the *Arabidopsis* circadian clock.," *Plant Cell*, vol. 24, no. February, pp. 428–43, 2012.

[230] T. Ohara, H. Fukuda, and I. T. Tokuda, "Phase Response of the Arabidopsis thaliana Circadian Clock to Light Pulses of Different Wavelengths," *J. Biol. Rhythms*, vol. 30, no. 2, pp. 95–103, 2015.

[231] K. Fogelmark and C. Troein, "Rethinking Transcriptional Activation in the *Arabidopsis* Circadian Clock," *PLoS Comput. Biol.*, vol. 10, no. 7, 2014.

[232] A. N. Dodd, J. Love, and A. A. R. Webb, "The plant clock shows its metal: circadian regulation of cytosolic free Ca^{2+}," *Trends Plant Sci.*, vol. 10, no. 1, pp. 15–21, Oct. 2017.

[233] J. Jin, "Mathematical Modeling of the Circadian Ca^{2+} Signaling Network," University of Cambridge, 2015.

[234] M. J. Gardner, K. E. Hubbard, C. T. Hotta, A. N. Dodd, and A. A. R. Webb, "How plants tell the time," *Biochem. J.*, vol. 397, no. Pt 1, pp. 15–24, Jul. 2006.

[235] C. R. McClung, "Plant circadian rhythms.," *Plant Cell*, vol. 18, no. 4, pp. 792–803, 2006.

[236] M. S. Robles, S. J. Humphrey, and M. Mann, "Phosphorylation Is a Central Mechanism for Circadian Control of Metabolism and Physiology," *Cell Metab.*, vol. 25, no. 1, pp. 118–127, Oct. 2017.

[237] J. L. Pruneda-Paz, G. Breton, A. Para, and S. A. Kay, "A functional genomics approach reveals CHE as a component of the *Arabidopsis* circadian clock.," *Science*, vol. 323, no. 5920, pp. 1481–5, Mar. 2009.

[238] D. Alabadí, M. J. Yanovsky, P. Más, S. L. Harmer, and S. A. Kay, "Critical Role for CCA1 and LHY in Maintaining Circadian Rhythmicity in *Arabidopsis*," *Curr. Biol.*, vol. 12, no. 9, pp. 757–761, Oct. 2017.

[239] T. Mizuno and N. Nakamichi, "Pseudo-Response Regulators (PRRs) or True Oscillator Components (TOCs)," *Plant Cell Physiol.*, vol. 46, no. 5, pp. 677–685, May 2005.

[240] E. M. Farré, S. L. Harmer, F. G. Harmon, M. J. Yanovsky, and S. A. Kay, "Overlapping and Distinct Roles of PRR7 and PRR9 in the *Arabidopsis* Circadian Clock," *Curr. Biol.*, vol. 15, no. 1, pp. 47–54, Oct. 2017.

[241] N. Nakamichi, M. Kita, S. Ito, T. Yamashino, and T. Mizuno, "PSEUDO-RESPONSE REGULATORS, PRR9, PRR7 and PRR5, Together Play Essential Roles Close to the Circadian Clock of *Arabidopsis thaliana*," *Plant Cell Physiol.*, vol. 46, no. 5, pp. 686–698, May 2005.

[242] H. G. McWatters, R. M. Bastow, A. Hall, and A. J. Millar, "The ELF3zeitnehmer regulates light signalling to the circadian clock," *Nature*, vol. 408, no. 6813, pp. 716–720, Dec. 2000.

[243] S. P. Hazen, T. F. Schultz, J. L. Pruneda-Paz, J. O. Borevitz, J. R. Ecker, and S. A. Kay, "LUX ARRHYTHMO encodes a Myb domain protein essential for circadian rhythms," *Proc. Natl. Acad. Sci. United States Am.*, vol. 102, no. 29, pp. 10387–10392, Jul. 2005.

[244] E. A. Kikis, R. Khanna, and P. H. Quail, "ELF4 is a phytochrome-regulated component of a negative-feedback loop involving the central oscillator components CCA1 and LHY.," *Plant J.*, vol. 44, no. 2, pp. 300–313, Oct. 2005.

[245] M. R. Doyle *et al.*, "The ELF4 gene controls circadian rhythms and flowering time in *Arabidopsis thaliana*," *Nature*, vol. 419, no. 6902, pp. 74–77, Sep. 2002.

[246] M. A. Jones, "Entrainment of the *Arabidopsis* Circadian Clock," *J. Plant Biol.*, vol. 52, no.

3, pp. 202–209, 2009.

[247] D. E. Somers, P. F. Devlin, and S. A. Kay, “Phytochromes and Cryptochromes in the Entrainment of the *Arabidopsis* Circadian Clock,” *Science*, vol. 282, no. 5393, p. 1488 LP-1490, Nov. 1998.

[248] Z. Y. Wang and E. M. Tobin, “Constitutive Expression of the CIRCADIAN CLOCK ASSOCIATED 1 (CCA1) Gene Disrupts Circadian Rhythms and Suppresses Its Own Expression,” *Cell*, vol. 93, no. 7, pp. 1207–1217, Oct. 2017.

[249] M. Ni, J. M. Tepperman, and P. H. Quail, “PIF3, a Phytochrome-Interacting Factor Necessary for Normal Photoinduced Signal Transduction, Is a Novel Basic Helix-Loop-Helix Protein,” *Cell*, vol. 95, no. 5, pp. 657–667, Oct. 2017.

[250] K. M. David, U. Armbruster, N. Tama, and J. Putterill, “*Arabidopsis* GIGANTEA protein is post-transcriptionally regulated by light and dark,” *FEBS Lett.*, vol. 580, no. 5, pp. 1193–1197, Feb. 2006.

[251] P. A. Salomé and C. R. McClung, “PSEUDO-RESPONSE REGULATOR 7 and 9 Are Partially Redundant Genes Essential for the Temperature Responsiveness of the *Arabidopsis* Circadian Clock,” *Plant Cell*, vol. 17, no. 3, pp. 791–803, Mar. 2005.

[252] N. A. Eckardt, “Temperature Compensation of the Circadian Clock: A Role for the Morning Loop,” *Plant Cell*, vol. 22, no. 11, p. 3506, Nov. 2010.

[253] J. KUSAKINA, P. D. GOULD, and A. HALL, “A fast circadian clock at high temperatures is a conserved feature across *Arabidopsis* accessions and likely to be important for vegetative yield,” *Plant. Cell Environ.*, vol. 37, no. 2, pp. 327–340, Feb. 2014.

[254] P. D. Gould *et al.*, “The Molecular Basis of Temperature Compensation in the *Arabidopsis* Circadian Clock,” *Plant Cell*, vol. 18, no. 5, pp. 1177–1187, May 2006.

[255] P. Suarez-Lopez, “CONSTANS mediates between the circadian clock and the control of flowering in *Arabidopsis*,” *Nature*, vol. 410. pp. 1116–1120, 2001.

[256] F. Valverde, A. Mouradov, W. Soppe, D. Ravenscroft, A. Samach, and G. Coupland, “Photoreceptor Regulation of CONSTANS Protein in Photoperiodic Flowering,” *Science*, vol. 303, no. 5660, p. 1003 LP-1006, Feb. 2004.

[257] T. Imaizumi and S. A. Kay, “Photoperiodic control of flowering: not only by coincidence,” *Trends Plant Sci.*, vol. 11, no. 11, pp. 550–558, 2006.

[258] A. N. Dodd, F. E. Belbin, A. Frank, and A. A. R. Webb, “Interactions between circadian clocks and photosynthesis for the temporal and spatial coordination of metabolism,” *Front. Plant Sci.*, vol. 6, p. 245, Apr. 2015.

[259] A. N Dodd, J. Kusakina, A. Hall, P. Gould, and M. Hanaoka, “The circadian regulation of photosynthesis,” *Photosynth. Res.*, vol. 119, Mar. 2013.

[260] M. J. Berridge, M. D. Bootman, and H. L. Roderick, “Calcium signalling: dynamics, homeostasis and remodelling.,” *Nat. Rev. Mol. Cell Biol.*, vol. 4, no. 7, pp. 517–29, Jul. 2003.

[261] G. J. Allen *et al.*, “A defined range of guard cell calcium oscillation parameters encodes stomatal movements,” *Nature*, vol. 411, no. 6841, pp. 1053–1057, Jun. 2001.

[262] A. M. Hetherington and C. Brownlee, “The generation of Ca2+ signals in plants,” *Annu. Rev. Plant Biol.*, vol. 55, no. 1, pp. 401–427, Apr. 2004.

[263] D. Sanders, J. Pelloux, C. Brownlee, and J. F. Harper, “Calcium at the Crossroads of Signaling,” *Plant Cell*, vol. 14, no. Suppl, pp. s401–s417, Mar. 2002.

[264] M. Ikeda *et al.*, “Circadian Dynamics of Cytosolic and Nuclear Ca^{2+} in Single Suprachiasmatic Nucleus Neurons,” *Neuron*, vol. 38, no. 2, pp. 253–263, Oct. 2017.

[265] A. N. Dodd *et al.*, “The *Arabidopsis* Circadian Clock Incorporates a cADPR-Based Feedback Loop,” *Science*, vol. 318, no. 5857, pp. 1789–1792, Dec. 2007.

[266] X. Xu *et al.*, “Distinct Light and Clock Modulation of Cytosolic Free Ca^{2+} Oscillations and Rhythmic CHLOROPHYLL A/B BINDING PROTEIN2 Promoter Activity in *Arabidopsis*,” *Plant Cell*, vol. 19, no. 11, p. 3474 LP-3490, Dec. 2007.

[267] R. M. Green and E. M. Tobin, “Loss of the circadian clock-associated protein 1 in *Arabidopsis* results in altered clock-regulated gene expression,” *Proc. Natl. Acad. Sci.*, vol. 96, no. 7, pp. 4176–4179, Mar. 1999.

[268] D. Alabadí, T. Oyama, M. J. Yanovsky, F. G. Harmon, P. Más, and S. A. Kay, “Reciprocal Regulation Between TOC1 and LHY/CCA1 Within the *Arabidopsis* Circadian Clock,” *Science*, vol. 293, no. 5531, p. 880 LP-883, Aug. 2001.

[269] L. Greengard and J. Strain, “THE FAST GAUSS TRANSFORM,” *Stat. Comput.*, vol. 12, no. 1, pp. 79–94, 1991.

[270] C. R. Dietrich and G. N. Newsam, “Fast and exact simulation of stationary gaussian processes through circulant embedding of the covariance matrix,” *SIAM J.SCI.COMPUT*, vol. 18, no. 4, pp. 1088–1107, 1997.

[271] I.Karatzas and S.Shreve, *Brownian Motion and Stochastic Calculus*. 1991.

[272] Z. Yue, “Dynamic Network Reconstruction in Systems Biology: Methods and Algorithms,” University of Luxembourg, 2018.