

# A Full Bayesian Approach to Sparse Network Inference using Heterogeneous Datasets

Junyang Jin, Ye Yuan, and Jorge Gonalves

**Abstract**—Network inference has been attracting increasing attention in several fields, notably systems biology and biomedicine. Indeed, one of our biggest challenges is to uncover and understand complex molecular networks behind cells and organisms. A network is mainly characterized by its topology and internal dynamics. In particular, sparse topology with stable dynamics are properties present in most real-world networks. Moreover, experiments typically measure a partial set of nodes. Linear systems have been used as approximations of complex nonlinear systems in a wide range of applications. While they exhibit simpler dynamics, they can easily model unmeasured nodes, e.g. via transfer functions, which is not possible for nonlinear systems. This paper explores these properties and considers general linear network models. It develops a method, based on reversible jump Markov chain Monte Carlo (RJMCMC), to estimate the confidence of all links, and produce the most likely topology without requiring any parameter tuning or choice of thresholds. Monte Carlo simulations indicate that our approach consistently produces more accurate networks compared with other state-of-the-art methods, including kernel based methods, nonlinear ODE based methods (iCheMA) and machine learning based methods (dynGENIE3). We show this is also true on a well known biological nonlinear model. The proposed method can be used in a wide range of applications, such as systems biology and biomedicine, and fault detection and diagnosis.

**Index Terms**—System Identification, Reversible Jump Markov Chain Monte Carlo, Dynamical Structure Function, Network Inference, Sparse Networks.

## I. INTRODUCTION

This paper addresses the inference of stable and sparse networks from data, with applications to both technological and biological systems. For example, communication systems are typically designed with sparse and stable structures to reduce energy consumption and to ensure long-term operation. In biology, most networks are inherently stable and sparse (molecules tend to bind to a very small number of other molecules). Moreover, experiments typically measure a partial set of nodes. For instance, in cell biology, experiments usually focus on a single type of molecules such as gene expression, protein or metabolites levels. Hence, the measured network can be considerably smaller than the complete network, since there is a large number of unmeasured nodes.

Whilst most practical networks are nonlinear, linear models have been widely used to approximate complex behaviours. While linear systems have simpler dynamics, they are considerably easier to model, analyse and control. Moreover,

certain *a priori* information, such as system stability, can be easily imposed. More importantly, linear systems can handle unmeasured nodes, which can be very hard for unknown nonlinear systems. To capture these properties, this paper considers a general model class known as dynamical structure function (DSF) to depict discrete-time linear, sparse networks with unmeasured nodes encoded via transfer functions.

Various methods have been developed to infer networks, either linear or nonlinear. These include iCheMA [1] and dynGENIE3 [2] for continuous-time systems, and GENIE3 [3] for discrete-time systems. iCheMA has been shown to outperform several state-of-the-art methods, such as hierarchical Bayesian regression (HBR), LASSO and elastic net through Monte Carlo simulations [1]. A more recent approach, dynGENIE3, has been compared with several methods in the literature [2]. Its minor variant, GENIE3, was the winner of a sub-challenge in DREAM4 [4]. Whilst these methods present good overall performance, they have several weaknesses: they cannot promote system stability when inferring linear systems and cannot produce the most likely topology without first choosing a threshold to select inferred links. The choice of thresholds can bias the results and typically requires prior knowledge of the network.

In recent years, kernel based non-parametric methods have been prevalent to identify linear systems [5]–[9] with several advantages in real-world applications. For example, they do not require estimating model complexity. More importantly, kernel based methods enforce system stability effectively by using proper kernel functions [10], [11]. Hyperparameters of kernels control system dynamics and network topology. Under the Bayesian paradigm, they are often estimated using empirical Bayes [12]–[16]. However, kernel based methods may not be ideal for topology detection due to suboptimal solutions. Model selection strategies (e.g. backward selection [14]) can be applied as a remedy. However, they have limitations: the confidence of inference cannot be evaluated accurately.

Monte Carlo techniques (MC) provide an alternative approximation inference [12]. They belong to stochastic approximations based on numerical sampling. Reversible jump Markov chain Monte Carlo (RJMCMC) is one of the MC approaches that was originally developed for Bayesian model selection [17]. RJMCMC has been applied in many research fields including optimization [18], [19], machine learning [20], [21], signal processing [22], and system identification [23], [24]. RJMCMC is able to draw samples from distributions whose random variables are of varying dimension. For a network, the dimension of model parameters depends on network topology: only true links are endowed with parameters.

Junyang Jin is with the Luxembourg Centre for Systems Biomedicine. Ye Yuan is with School of Automation, Huazhong University of Science and Technology. Jorge Gonalves is with the Department of Engineering, University of Cambridge and the Luxembourg Centre for Systems Biomedicine. Corresponding author: Ye Yuan (ye.yuan@outlook.com).

Hence, RJMCMC represents a promising method for network inference.

This paper applies Gaussian processes and RJMCMC to infer sparse networks. DSFs are used to describe general linear networks, including unmeasured nodes encoded via transfer functions. As a non-parametric method, Gaussian processes are applied to impose stable impulse responses of networks. RJMCMC is adopted to explore the resulting Bayesian model whose sampling space consists of multiple subspaces of different dimensionality. By traversing these subspaces, RJMCMC provides a highly efficient way to infer system dynamics and to detect network topology. In particular, the effect of Automatic Relevance Determination (ARD) introduced by hyperparameters is maximally activated by the merit of RJMCMC, thus encouraging sparse topologies. Monte Carlo simulations indicate that our method outperforms state-of-the-art methods, including kernel based methods, iCheMA, dynGINIE3 and GENIE3. While our method is derived for discrete-time systems, we show that it can also have better performance than these methods when applied to continuous-time systems, as illustrated on a well-known nonlinear continuous-time biological network.

The paper is organized as follows. Section II introduces dynamical structure functions and formulates full Bayesian models. Section III overviews MH-within-PCG samplers and RJMCMC. Section IV discusses network inference using RJMCMC. Section V compares the method with other approaches via Monte Carlo simulations. Finally, Section VI concludes and discusses further development in this field.

*Notation:* The notation in this paper is standard.  $I_m$  denotes the  $m \times m$  identity matrix. For  $L \in \mathbb{R}^{n \times n}$ ,  $\text{diag}\{L\}$  denotes a vector which consists of diagonal elements of matrix  $L$ .  $[L]_{ij}$  presents the  $ij$ th entry and  $L(:, i : j)$  the columns from  $i$  to  $j$ .  $\text{blkdiag}\{L_1, \dots, L_n\}$  is a block diagonal matrix. For  $l \in \mathbb{R}^n$ ,  $\text{diag}\{l\}$  denotes a diagonal matrix whose diagonal elements come from vector  $l$ .  $[l]_{ij}$  denotes the  $j$ th element of the  $i$ th group of  $l$ .  $l \geq 0$  means each element of the vector is non-negative.  $y(t_1 : t_2)$  denotes a row vector  $[y(t_1) \ y(t_1 + 1) \ \dots \ y(t_2)]$ .

## II. MODEL SPECIFICATION

### A. The dynamical structure function

This paper considers linear networks with  $p$  measured nodes and unknown number of hidden nodes. These networks can be described by Dynamical Structure Functions (DSF) as follows [25]–[27]:

$$Y = Q(q; \theta)Y + P(q; \theta)U + H(q; \theta)E. \quad (1)$$

where  $q$  denotes the time shift operator ( $y(t + 1) = qy(t)$ ). Outputs  $Y \in \mathbb{R}^p$  are the measured nodes,  $U \in \mathbb{R}^m$  are the inputs,  $E \in \mathbb{R}^q$  are i.i.d. Gaussian noise, and  $\theta$  are model parameters.

$Q$ ,  $P$  and  $H$  are transfer matrices, where each element is a transfer function encoding for unmeasured nodes. Matrix  $Q$  represents the connectivity among measured nodes. Its transfer functions are strictly proper and its diagonal elements are zero. Matrices  $P$  and  $H$  relate inputs and process noise to nodes,

respectively. The transfer functions of matrix  $P$  are strictly proper whilst those of matrix  $H$  are proper. The topology of the network is reflected by the location of the zeros in  $Q$ . For example, if  $[Q]_{ij}$  is zero, the  $j$ th node does not directly control the  $i$ th node. Let  $\mathcal{M}_k$  denote model structures (network topology) and  $M_k$  represent the corresponding number of nonzero links. In particular,  $\mathcal{M}_1$  represents the fully-connected topology. In general, the number of hidden (non-measured) states is unknown, which means that the order of each transfer function is also unknown.

The input-output map of a network is deduced based on the DSF as follows:

$$Y = G_u U + G_e E. \quad (2)$$

where  $G_u = (I - Q)^{-1}P$  and  $G_e = (I - Q)^{-1}H$ . Identifiability of the network depends on whether an input-output map is associated to a unique DSF. Ensuring that an inference problem is well-posed requires additional constraints on the structure of transfer matrices.

*Proposition 1 (Identifiability of DSF networks [25]):* Given a  $p \times (m + q)$  transfer matrix  $G = [G_u, G_e]$ , the corresponding DSF is unique if and only if  $p - 1$  elements in each column of  $[Q, P, H]'$  are known, which uniquely specifies the component of  $(Q, P, H)$  in the null space of  $[G', I]$ .

A sufficient condition for network identifiability is that matrix  $H$  is diagonal, so that  $p - 1$  elements in each column of  $[Q, P, H]'$  are known and equal to zero. Hence, we make following assumptions.

*Assumption 1:* Noise matrix  $H$  is diagonal, monic ( $\lim_{q \rightarrow \infty} H = I$ ) and minimal phase.

*Assumption 2:* Input-output maps  $G_u$  and  $G_e$  are stable. Transfer matrices,  $Q$  and  $P$  are sparse and stable where each transfer function is stable.

### B. The likelihood distribution

After simple manipulations, the DSF in (1) can be reformulated as:

$$Y = F_y(q; \theta)Y + F_u(q; \theta)U + E. \quad (3)$$

where

$$\begin{aligned} F_u(q; \theta) &= H^{-1}P. \\ F_y(q; \theta) &= I - H^{-1}(I - Q). \end{aligned} \quad (4)$$

From the above assumptions, it follows that transfer functions  $F_u$  and  $F_y$  are stable, and they have the same zero structures as  $P$  and  $Q$ , respectively (since  $H$  is diagonal).

Identifying transfer functions of model (3) is non-trivial: estimating the order of transfer functions requires an exhaustive search of all possibilities, which is computationally prohibitive for large-scale networks. Additionally, imposing stable transfer functions can be problematic. The number of constraints for stable poles depends on the order of transfer functions and grows with respect to the scale of networks. A possible solution to these problems is to express model (3) non-parametrically. This way, selection of model complexity is avoided and, more importantly, system stability can be promoted effectively.

The dynamical system for the  $i$ th target node can be formulated as:

$$y_i(t) = \sum_{j=1}^p \sum_{k=1}^{\infty} h_{ij}^y(k) y_j(t-k) + \sum_{j=1}^m \sum_{k=1}^{\infty} h_{ij}^u(k) u_j(t-k) + e_i(t). \quad (5)$$

where  $h_{ij}^y$  and  $h_{ij}^u$  are the impulse responses of transfer functions  $[F_y]_{ij}$  and  $[F_u]_{ij}$ , respectively.  $e_i(t)$  is i.i.d. Gaussian noise. Since the impulse responses are stable, they decay exponentially fast. For the implementation purpose, they can be truncated after sample time  $T$ , which is set sufficiently large to catch major dynamics of the impulse responses (i.e.  $|h(k)| \approx 0$  for  $k \geq T$ ).

Assume the availability of time series data collected from discrete time indices 1 to  $N$  for each node and input. For the  $i$ th target node with  $\mathcal{M}_1$  and a single experiment, we define the following matrices and vectors (for other possible model structures, the corresponding terms are defined similarly):

$$Y = \begin{bmatrix} y_i(N) \\ \vdots \\ y_i(T+1) \end{bmatrix}, W = \begin{bmatrix} w_1 \\ \vdots \\ w_{p+m} \end{bmatrix},$$

$$\Phi = [\Phi_y \quad \Phi_u],$$

$$\Phi_y = \begin{bmatrix} y_1(N-1:N-T) & \cdots & y_p(N-1:N-T) \\ \vdots & \ddots & \vdots \\ y_1(T:1) & \cdots & y_p(T:1) \end{bmatrix},$$

$$\Phi_u = \begin{bmatrix} u_1(N-1:N-T) & \cdots & u_m(N-1:N-T) \\ \vdots & \ddots & \vdots \\ u_1(T:1) & \cdots & u_m(T:1) \end{bmatrix},$$

$$\sigma = E\{e_i(t)^2\}. \quad (6)$$

where  $Y \in R^{N-T}$  are time series of the  $i$ th node.  $W \in R^{T(p+m)}$  contain  $p+m$  groups of impulse responses, each of which corresponds to a transfer function of  $F_y$  or  $F_u$ .  $\Phi \in R^{(N-T) \times T(p+m)}$  include time series of all the nodes and inputs.  $\sigma$  is the noise variance. Note that the dimensions of these quantities vary with respect to model structures. For example, if node  $j$  does not control node  $i$ ,  $w_j$  and  $\Phi(:, T(j-1)+1 : Tj)$  must be removed from the corresponding vector and matrix. Hence, for model structure  $\mathcal{M}_k$ ,  $Y \in R^{N-T}$ ,  $W \in R^{TM_k}$  and  $\Phi \in R^{(N-T) \times TM_k}$ .

Based on Bayes' rules, the likelihood distribution of the  $i$ th target node with  $\mathcal{M}_k$  is:

$$p(Y|W, \sigma, \mathcal{M}_k) = (2\pi\sigma)^{-\frac{N-T}{2}} \exp\left\{-\frac{1}{2\sigma} \|Y - \Phi W\|_2^2\right\}. \quad (7)$$

### C. The prior distributions

Full Bayesian treatment deploys prior distributions for each random quantity to build up a hierarchical structure. Under the Bayesian paradigm, impulse responses are assumed to be independent Gaussian processes [14]. To impose stable impulse responses, covariance functions (kernel functions) must

be chosen carefully. It has been shown that Tuned/Correlated kernel (TC), Diagonal/Correlated kernel (DC) and second order stable spline kernel (SS) are all capable of characterizing reproducing kernel Hilbert spaces (RKHS) for stable impulse responses. They have been frequently applied in the system identification community [10], [28]. Hence, these three kernel functions are all considered in our framework. The prior distribution for  $W$  is:

$$p(W|\lambda, \beta, \mathcal{M}_k) = \prod_{i=1}^{M_k} \mathcal{N}(w_i|0, \lambda_i K_i). \quad (8)$$

where  $K_i \in R^{T \times T}$ ,  $\lambda = [\lambda_1, \dots, \lambda_{M_k}]'$  and  $\beta = [\beta_1, \dots, \beta_{M_k}]'$ . Note that for DC kernel,  $\beta_i$  is a row vector consisting of two elements whilst it is a scalar for TC and SS kernels.

$$[K_i]_{ts} = k(t, s; \beta_i), \quad \lambda_i \geq 0,$$

$$k_{TC}(t, s; \beta_i) = \beta_i^{max(t,s)}, \quad \beta_i \in (0, 1),$$

$$k_{DC}(t, s; \beta_i) = \beta_{i1}^{\frac{(t+s)}{2}} \beta_{i2}^{|t-s|}, \quad \beta_{i1} \in (0, 1), \quad \beta_{i2} \in (-1, 1),$$

$$k_{SS}(t, s; \beta_i) = \frac{\beta_i^{t+s+max(t,s)}}{2} - \frac{\beta_i^{3max(t,s)}}{6}, \quad \beta_i \in (0, 1). \quad (9)$$

Here,  $\beta$  are hyperparameters of the kernel functions, which control the decaying rate of impulse responses [11], while  $\lambda$  are scale variables of the kernel functions. They play the role of Automatic Relevance Determination (ARD) parameters that control sparsity [14]. If  $\lambda_i$  approaches zero, the corresponding impulse responses  $w_i$  are forced to zero, meaning they can be removed from the model.

Since  $\sigma$  is non-negative, the Inverse-Gamma distribution is used as its conjugate prior. Without specific preference on  $\sigma$ , parameters  $a_0$  and  $b_0$  of the distribution are set to 0.001, resulting in a non-informative prior:

$$p(\sigma; a_0, b_0) = IG(\sigma; a_0, b_0) = \frac{b_0^{a_0}}{\Gamma(a_0)} \sigma^{-a_0-1} e^{-\frac{b_0}{\sigma}}, \quad (10)$$

where  $\Gamma(\cdot)$  is the gamma function.

Instead of introducing equal probabilities for network topologies, the prior distribution for  $\mathcal{M}_k$  depends on the number of links,  $M_k$ . The cardinality of the set of all possible topologies equates to  $|\mathcal{M}| = \sum_{i=0}^{M_1-1} \mathcal{C}(M_1-1, i)$  where  $\mathcal{C}(n, m) = \frac{n(n-1)\cdots(n-m+1)}{m!}$ . The prior of  $\mathcal{M}_k$  is a minor variant of the truncated Poisson distribution:

$$p(\mathcal{M}_k|\alpha) = \frac{\alpha^{M_k} (M_k!)^{-1}}{\sum_{i=1}^{|\mathcal{M}|} \alpha^{M_i} (M_i!)^{-1}}. \quad (11)$$

where  $\alpha$  is the rate parameter of the Poisson distribution. Distribution (11) favours sparse topologies since higher probability is assigned to topologies with lower number of links. However, different topologies are equally distributed if they have the same number of links.

Hyperpriors are assigned to hyperparameters to complete the hierarchy. For non-negative hyperparameter  $\lambda_i$ , the Inverse-Gamma distribution is applied as the conjugate prior:  $p(\lambda_i; a_1, b_1) = IG(\lambda_i; a_1, b_1)$ . To impose sparsity, we set  $a_1 = 2$  and  $b_1 = 1$  so that the distribution has infinite variance and puts most of weights over small values.

For hyperparameter  $\beta_i$ , the Uniform distribution is employed as its prior:

$$\begin{aligned} TC/SS : p(\beta_i) &= 1, \beta_i \in (0, 1), \\ DC : p(\beta_i) &= \frac{1}{2}, \beta_{i1} \in (0, 1), \beta_{i2} \in (-1, 1). \end{aligned} \quad (12)$$

Finally, the Gamma distribution is assigned to hyperparameter  $\alpha$ :

$$p(\alpha; a_2, b_2) = \text{Gamma}(\alpha; a_2, b_2) = \frac{b_2^{a_2}}{\Gamma(a_2)} \alpha^{a_2-1} e^{-b_2 \alpha}. \quad (13)$$

To promote sparse topologies, we set  $a_2 = 0.1$  and  $b_2 = 1$  so that  $p(\alpha; a_2, b_2)$  approaches infinity at  $\alpha = 0$ . However, since parameters  $a_2$  and  $b_2$  are deep in the hierarchy, they have little impact on the model.

#### D. The posterior distributions

Consider a heterogeneous dataset containing  $L$  independent time series of the target network, which are collected under different experimental conditions. This paper assumes that the internal dynamics of the network vary with experimental conditions whilst the network topology remains unchanged. Hence, impulse responses under different experimental conditions are independently distributed and the dimension of random variables is identical for all experiments. Note that our framework can be extended to other cases easily. For example, if some genes of the network are knocked out, their corresponding impulse responses are consistently removed from the model. In practice, the proposed model can be freely adjusted according to the experimental condition.

Based on Bayes' rules and after completing squares, the posterior distribution of the DSF can be written as:

$$\begin{aligned} & p(\mathcal{M}_k, W, \beta, \lambda, \sigma, \alpha | Y) \\ & \propto p(Y | W, \sigma, \mathcal{M}_k) \left[ \prod_{j=1}^L p(W_j | \beta, \lambda, \sigma, \mathcal{M}_k) \right] p(\sigma) \\ & \quad p(\beta | \mathcal{M}_k) p(\lambda | \mathcal{M}_k) p(\mathcal{M}_k | \alpha) p(\alpha) \\ & \propto \left[ \prod_{j=1}^L (2\pi\sigma_j)^{-\frac{N_j-T}{2}} \exp \left\{ -\frac{1}{2} Y_j' (\sigma_j I + \Phi_j \Lambda K \Phi_j')^{-1} Y_j \right\} \right] \\ & \quad \times |2\pi \Lambda K|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (W_j - \mu_j)' \Sigma_j^{-1} (W_j - \mu_j) \right\} \\ & \quad \times \sigma_j^{-a_0-1} \exp \left\{ -\frac{b_0}{\sigma_j} \right\} \alpha^{a_2-1} \exp\{-b_2 \alpha\} \\ & \quad \times \frac{\alpha^{M_k} (M_k!)^{-1}}{\sum_{i=1}^{|\mathcal{M}|} \alpha^{M_i} (M_i!)^{-1}} \prod_{i=1}^{M_k} \frac{b_1^{a_1}}{2\Gamma(a_1)} \lambda_i^{-a_1-1} \exp \left\{ -\frac{b_1}{\lambda_i} \right\}. \end{aligned} \quad (14)$$

where subscript  $j$  denotes the index of experiments. Note that hyperparameters of the kernels are shared by different experiments, implying the same prior knowledge for system dynamics. The number of measurements of the  $j$ th experiment is  $N_j$  and

$$\begin{aligned} K &= \text{blkdiag}\{K_1, \dots, K_{M_k}\}, \Lambda = \text{diag}\{\lambda\} \otimes I_T, \\ \Sigma_j^{-1} &= \frac{1}{\sigma_j} \Phi_j' \Phi_j + (\Lambda K)^{-1}, \mu_j = \frac{1}{\sigma_j} \Sigma_j \Phi_j' Y_j. \end{aligned} \quad (15)$$

According to the full Bayesian model in (14), the conditional posterior distributions of the random variables are:

$$\begin{aligned} p(W | \beta, \lambda, \sigma, \alpha, \mathcal{M}_k, Y) &= \prod_{j=1}^L \mathcal{N}(W_j | \mu_j, \Sigma_j), \\ p(\sigma | W, \beta, \lambda, \alpha, \mathcal{M}_k, Y) &= \prod_{j=1}^L IG(\sigma_j; a_{\sigma_j}, b_{\sigma_j}), \\ p(\alpha | W, \beta, \lambda, \sigma, \mathcal{M}_k, Y) &\propto \frac{\alpha^{a_2-1+M_k} (M_k!)^{-1}}{\sum_{i=1}^{|\mathcal{M}|} \alpha^{M_i} (M_i!)^{-1}} e^{-b_2 \alpha}, \end{aligned} \quad (16)$$

where

$$a_{\sigma_j} = a_0 + \frac{N_j - T}{2}, \quad b_{\sigma_j} = b_0 + \frac{\|Y_j - \Phi_j W_j\|_2^2}{2}. \quad (17)$$

By marginalizing  $W$  out from the full Bayesian model, the reduced joint posterior distribution of  $\mathcal{M}_k, \beta$  and  $\lambda$  is given by:

$$\begin{aligned} & p(\beta, \lambda, \mathcal{M}_k | \sigma, \alpha, Y) \\ & \propto \prod_{j=1}^L |\sigma_j I + \Phi_j \Lambda K \Phi_j'|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} Y_j' (\sigma_j I + \Phi_j \Lambda K \Phi_j')^{-1} Y_j \right\} \\ & \quad \times \frac{\alpha^{M_k} (M_k!)^{-1}}{\sum_{i=1}^{|\mathcal{M}|} \alpha^{M_i} (M_i!)^{-1}} \prod_{i=1}^{M_k} \frac{b_1^{a_1}}{2\Gamma(a_1)} \lambda_i^{-a_1-1} \exp \left\{ -\frac{b_1}{\lambda_i} \right\}. \end{aligned} \quad (18)$$

### III. OVERVIEW OF MARKOV CHAIN MONTE CARLO

#### A. MH-within-PCG Sampler

Markov Chain Monte Carlo (MCMC) methods have been widely applied to draw samples from probabilistic models. The samples consist of a Markov chain that asymptotically distributes as the target distribution.

Gibbs sampling and Metropolis-Hastings methods (MH) are two typical MCMC techniques [13]. They have their own strength and weakness. Gibbs samplers have a simple structure but require full expressions of conditional distributions. MH samplers can sample from a distribution known up to a constant but their design is more involved. In practice, Gibbs and MH samplers are often modified and combined accordingly to handle complex distributions [29], [30].

MH-within-Gibbs samplers are widely used in many applications. They combine the scheme of Gibbs and MH samplers. If the conditional distributions of certain sampling steps of a Gibbs sampler are only known up to a normalization constant, one can replace these steps with MH sampling, leading to a hybrid sampler [31]. Often, some random variables are marginalized out from distributions to improve convergence properties of samplers [32]. Gibbs and MH-within-Gibbs samplers with such modification are called partially collapsed Gibbs samplers (PCG) and Metropolis-Hastings within partially collapsed Gibbs samplers (MH-within-PCG), respectively [31].

Designing PCG samplers is non-trivial. Some rules must be obeyed to marginalize conditional distributions of Gibbs samplers, otherwise invariant distributions of Markov chains may be changed [31]. However, this issue was not sufficiently aware of in much of the previous research. In general, steps called marginalization, permutation and trimming are executed

in sequence to reduce the number of conditioned random variables [32]. Marginalization means to move components of Markov states from being conditioned on to being sampled. For example, the sampling step of  $p(x|y, z)$  can be replaced with that of  $p(x, y|z)$ . Permutation means to switch the order of sampling steps. Finally, trimming is used to discard a subset of random variables that are not conditioned on in the next step. For example,  $p(x, y|z)$  can be replaced with  $p(x|z)$  if its sampling step is followed by that of  $p(y|x, z)$ . However, such a replacement is invalid if  $p(z|x, y)$  is sampled from in the next step.

It is important to realize that sampling steps of PCG samplers cannot be replaced with their MH counterparts directly. MH-within-PCG samplers must be derived from MH-with-Gibbs samplers following the similar rules of PCG [31]. Full MH steps of MH-with-Gibbs samplers can be replaced by reduced MH steps only if a direct draw from the full distribution of the reduced quantities follows up immediately [31]. For example, MH sampling of  $p(x|y, z)$  in a MH-within-Gibbs sampler can be replaced with MH sampling of  $p(x|z)$ , followed immediately by sampling from  $p(y|x, z)$ .

### B. Reversible Jump Markov Chain Monte Carlo

Reversible Jump Markov Chain Monte Carlo (RJCMCMC) was designed to explore distributions whose random variables have varying dimension [33], [34]. Whilst normal MCMC schemes can only explore a single sampling space, RJCMCMC is able to jump between parameter subspaces of different dimensions.

Consider a countable collection of Bayesian models  $\{\mathcal{M}_k, k \in \mathbb{Z}^+\}$ . Each model is associated with a parameter vector  $\theta_k \in \mathbb{R}^{d_k}$ , where  $d_k$  may differ from model to model. The random variable to be sampled is  $x = (k, \theta_k)$  which lies in the sampling space  $S = \bigcup_{k \in \mathbb{Z}^+} S_k$ . Sampling space  $S$  is composed of parameter subspaces of different dimensionality:  $S_k = \{k\} \times \mathbb{R}^{d_k}$ .

Suppose  $p(x)$  is the probability density function of interest. RJCMCMC produces a reversible Markov chain  $\{X^t, t \in \mathbb{Z}^+\}$ , regarding  $p(x)$  as the invariant distribution. Each Markov state  $X^t$  consists of two components,  $k^t$  and  $\theta_k^t$  where  $k^t$  is the model index and  $\theta_k^t$  is the corresponding model parameter. To traverse across the parameter subspaces, different types of moves are proposed in RJCMCMC, among which only one move is executed per iteration. Proposal distributions are designed for random variables and acceptance probability is calculated so that 'detailed balance' is achieved among different move types. The resulting transition distribution of the Markov chain is the mixing of that of all moves.

Let  $(k, \theta)$  be the current state  $X^t$  of the Markov chain where  $\theta \in \mathbb{R}^{d_k}$ . Based on the available moves, the probability to jump from the current model  $k$  to the next one  $k'$  is  $p_{kk'}$ , where  $\sum_{k'} p_{kk'} = 1$ . 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. Given the proposed  $k'$ ,  $\theta' \in \mathbb{R}^{d_{k'}}$  is generated as the proposal for the model parameter. One way to generate  $\theta'$  is to first produce a random quantity  $U$  with the probability density  $q_{kk'}(u|\theta)$  and

then map  $\theta$  and  $U$  to  $\mathbb{R}^{d_{k'}}$ . As a result,  $\theta' = g_{1kk'}(\theta, U)$  where  $U \in \mathbb{R}^{d_{kk'}}$  and  $g_{1kk'} : \mathbb{R}^{d_k + d_{kk'}} \rightarrow \mathbb{R}^{d_{k'}}$  is a deterministic map [34]. The proposal  $X^p = (k', \theta')$  is then accepted with probability  $A_{kk'}(\theta'|\theta)$ . If accepted,  $X^{t+1} = X^p$ , otherwise  $X_{t+1} = X_t$ .

For the move from  $(k, \theta)$  to  $(k', \theta')$  and the move in the opposite direction, their corresponding proposals,  $(\theta, U)$  and  $(\theta', U')$  must have equal dimension. This is called dimension matching:  $d_k + d_{kk'} = d_{k'} + d_{k'k}$  [17]. In addition, there must exist a deterministic map  $g_{2kk'} : \mathbb{R}^{d_{k'} + d_{kk'}} \rightarrow \mathbb{R}^{d_{kk'}}$  such that  $(\theta', U') = g_{kk'}(\theta, U) = (g_{1kk'}(\theta, U), g_{2kk'}(\theta, U))$  where map  $g_{kk'}$  is bijective and differentiable [34].

Finally, to achieve 'detailed balance', the following equation must be satisfied [34]:

$$\begin{aligned} & p(x)p_{kk'}q_{kk'}(U|\theta)A_{kk'} \\ &= p(x')p_{k'k}q_{k'k}(U'|\theta')A_{k'k} \left| \frac{\partial g_{kk'}(\theta, U)}{\partial \theta \partial U} \right|. \end{aligned} \quad (19)$$

where

$$\theta' = g_{1kk'}(\theta, U) \text{ and } U' = g_{2kk'}(\theta, U). \quad (20)$$

As a result, the acceptance probability equates to:

$$\begin{aligned} & A_{kk'}(\theta'|\theta) \\ &= \min \left\{ 1, \frac{p(x')p_{k'k}q_{k'k}(U'|\theta')}{p(x)p_{kk'}q_{kk'}(U|\theta)} \left| \frac{\partial g_{kk'}(\theta, U)}{\partial \theta \partial U} \right| \right\}. \end{aligned} \quad (21)$$

## IV. NETWORK INFERENCE USING RJCMCMC

### A. Sampler with the fixed topology

With the full Bayesian model, we are interested in the marginal posterior distribution of network topologies,  $p(\mathcal{M}_k|Y)$ , by which we can determine the most likely network topology. Given the network topology, impulse responses and noise variance are estimated as their expectation values. However, distributions  $p(\mathcal{M}_k|Y)$ ,  $p(W|\mathcal{M}_k, Y)$  and  $p(\sigma|\mathcal{M}_k, Y)$  are intractable since one needs to perform high-dimensional integrals of the Bayesian model in (14). To solve the problem, numerical sampling methods are applied in our framework.

To begin with, assume that the topology of the target network is known *a priori* (e.g.  $\mathcal{M}_k$ ). Since the dimension of random variables is fixed, traditional MCMC methods are sufficient to draw samples from the distribution. To improve convergence properties, random variables  $W$  are marginalized out in certain sampling steps. The resulting MH-within-PCG sampler is designed following the rules of marginalization, permutation and trimming. The sampler is further modified to explore the network with unknown topology.

Gibbs sampling is accepted to construct the basic sampler (Sampler 1). Random variables are updated in sequence as follows.

---

#### **Sampler 1:** Blocked Gibbs sampler

---

- 1: Sample  $p(W^{t+1}|\beta^t, \lambda^t, \sigma^t, \alpha^t, \mathcal{M}_k, Y)$
  - 2: Sample  $p(\beta^{t+1}, \lambda^{t+1}|W^{t+1}, \sigma^t, \alpha^t, \mathcal{M}_k, Y)$
  - 3: Sample  $p(\sigma^{t+1}|W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \alpha^t, \mathcal{M}_k, Y)$
  - 4: Sample  $p(\alpha^{t+1}|W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \sigma^{t+1}, \mathcal{M}_k, Y)$
- 

Since the distributions of steps 2 and 4 are known up to a normalization constant, these two sampling steps should be

implemented using MH methods, leading to a MH-within-Gibbs sampler (Sampler 2).

---

**Sampler 2: MH-within-Gibbs sampler**

---

- 1: Sample  $p(W^{t+1}|\beta^t, \lambda^t, \sigma^t, \alpha^t, \mathcal{M}_k, Y)$
  - 2: Sample  $p(\beta^{t+1}, \lambda^{t+1}|W^{t+1}, \sigma^t, \alpha^t, \mathcal{M}_k, Y)$  using MH
  - 3: Sample  $p(\sigma^{t+1}|W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \alpha^t, \mathcal{M}_k, Y)$
  - 4: Sample  $p(\alpha^{t+1}|W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \sigma^{t+1}, \mathcal{M}_k, Y)$  using MH
- 

As indicated in (18), one can marginalize  $W$  out from distribution  $p(\beta, \lambda|W, \sigma, \alpha, \mathcal{M}_k, Y)$  of step 2 in Sampler 2. According to the rule of marginalization, the reduced sampling step is followed immediately by a direct draw from the conditional distribution of  $W$ , resulting in a MH-within-PCG sampler (Sampler 3). The quantity that is not conditioned on in the next step is labelled with an asterisk.

---

**Sampler 3: Marginalization**

---

- 1: Sample  $p(W^*|\beta^t, \lambda^t, \sigma^t, \alpha^t, \mathcal{M}_k, Y)$
  - 2: Sample  $p(\beta^{t+1}, \lambda^{t+1}|\sigma^t, \alpha^t, \mathcal{M}_k, Y)$  using MH
  - 3: Sample  $p(W^{t+1}|\beta^{t+1}, \lambda^{t+1}, \sigma^t, \alpha^t, \mathcal{M}_k, Y)$
  - 4: Sample  $p(\sigma^{t+1}|W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \alpha^t, \mathcal{M}_k, Y)$
  - 5: Sample  $p(\alpha^{t+1}|W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \sigma^{t+1}, \mathcal{M}_k, Y)$  using MH
- 

To further simply Sampler 3, permutation and trimming are applied. Note that in order to maintain the invariant distribution, steps 2 and 3 can neither be separated nor swapped. Since the sampled  $W$  in step 1 is not conditioned on in step 2, step 1 can be removed from the sampler. The resulting sampler is presented in Sampler 4.

---

**Sampler 4: Permutation and Trimming**

---

- 1: Sample  $p(\beta^{t+1}, \lambda^{t+1}|\sigma^t, \alpha^t, \mathcal{M}_k, Y)$  using MH
  - 2: Sample  $p(W^{t+1}|\beta^{t+1}, \lambda^{t+1}, \sigma^t, \alpha^t, \mathcal{M}_k, Y)$
  - 3: Sample  $p(\sigma^{t+1}|W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \alpha^t, \mathcal{M}_k, Y)$
  - 4: Sample  $p(\alpha^{t+1}|W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \sigma^{t+1}, \mathcal{M}_k, Y)$  using MH
- 

The sampling steps of Sampler 4 can be rearranged in many other ways. For example, they can be swapped as  $4 \rightarrow 3 \rightarrow 1 \rightarrow 2$ . However, not all the arrangements are valid. For example, sequence  $2 \rightarrow 1 \rightarrow 3 \rightarrow 4$  derived from trimming out step 3 in Sampler 3 is incorrect because it violates the rule of trimming. Hence, it is important to realize that sampling steps of PCG samplers cannot be replaced with their MH counterparts trivially.

According to (16), steps 2 and 3 in Sampler 4 can be implemented easily. Only steps 1 and 4 require further discussion. Since these two steps employ MH approaches, proposal distributions for Markov states are first designed to produce candidate samples.

For step 1, since  $\lambda$  are non-negative, the truncated Gaussian distribution is adopted to draw proposals. For the Gaussian distribution of  $\theta$  with mean  $\mu_0$  and variance  $\sigma_0$ , its truncated probability density function on  $(l, u)$  is:

$$p_{\mathcal{N}}(\theta; \mu_0, \sigma_0, l, u) = \frac{f\left(\frac{\theta - \mu_0}{\sigma_0}\right)}{\sigma_0 \left[ F\left(\frac{u - \mu_0}{\sigma_0}\right) - F\left(\frac{l - \mu_0}{\sigma_0}\right) \right]}, \quad (22)$$

where

$$\begin{aligned} f(x) &= \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, \\ F(x) &= \frac{1}{2} \left[ 1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right) \right], \\ \operatorname{erf}(x) &= \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \end{aligned} \quad (23)$$

The proposal distribution for  $\lambda$  is  $q(\lambda|\lambda^t) = \prod_{i=1}^{M_k} p_{\mathcal{N}}(\lambda_i; \lambda_i^t, 0.05, 0, +\infty)$ . In order to avoid the high rejection rate, the proposed  $\lambda^p$  only deviate from the current state  $\lambda^t$  with a small variance.

For hyperparameters  $\beta$ , the proposal of each element is drawn from the following distribution independently. For a random variable  $\theta \in (l, u)$  with the expected value  $\bar{\theta}$ , its probability density function is as follows:

$$p_U(\theta; \bar{\theta}, l, u, \varepsilon) = \begin{cases} U(\bar{\theta} - \frac{\varepsilon}{2}, \bar{\theta} + \frac{\varepsilon}{2}) & l + \frac{\varepsilon}{2} < \bar{\theta} < u - \frac{\varepsilon}{2} \\ U(l, l + \varepsilon) & \bar{\theta} \leq l + \frac{\varepsilon}{2} \\ U(u - \varepsilon, u) & \bar{\theta} \geq u - \frac{\varepsilon}{2} \end{cases}. \quad (24)$$

where  $U(a, b)$  is the uniform distribution on  $(a, b)$ .  $\varepsilon$  is the selection window for sampling. Hence, the proposal distribution for  $\beta_i$  is:

$$\begin{aligned} TC/SS : q(\beta_i|\beta_i^t) &= p_U(\beta_i; \beta_i^t, 0, 1, 0.1), \\ DC : q(\beta_i|\beta_i^t) &= p_U(\beta_{i1}; \beta_{i1}^t, 0, 1, 0.1) p_U(\beta_{i2}; \beta_{i2}^t, -1, 1, 0.1). \end{aligned} \quad (25)$$

According to 'detailed balance', the acceptance probability for step 1 is calculated as follows:

$$\begin{aligned} A_U(\beta^p, \lambda^p|\beta^t, \lambda^t) &= \min \left\{ 1, \frac{p(\beta^p, \lambda^p|\sigma^t, \alpha^t, \mathcal{M}_k, Y) q(\beta^t|\beta^p) q(\lambda^t|\lambda^p)}{p(\beta^t, \lambda^t|\sigma^t, \alpha^t, \mathcal{M}_k, Y) q(\beta^p|\beta^t) q(\lambda^p|\lambda^t)} \right\} \\ &= \min \{ 1, r_U(\beta^p, \lambda^p|\beta^t, \lambda^t) \}. \end{aligned} \quad (26)$$

where

$$\begin{aligned} r_U(\beta^p, \lambda^p|\beta^t, \lambda^t) &= \frac{\prod_{j=1}^L \exp\{-\frac{1}{2} Y_j'(\sigma_j^t I + \Phi_j \Lambda^p K^p \Phi_j')^{-1} Y_j\} |\sigma_j I + \Phi_j \Lambda^p K^p \Phi_j'|^{-\frac{1}{2}}}{\prod_{j=1}^L \exp\{-\frac{1}{2} Y_j'(\sigma_j^t I + \Phi_j \Lambda^t K^t \Phi_j')^{-1} Y_j\} |\sigma_j I + \Phi_j \Lambda^t K^t \Phi_j'|^{-\frac{1}{2}}} \\ &\times \prod_{i=1}^{M_k} \left( \frac{\lambda_i^p}{\lambda_i^t} \right)^{-a_i - 1} \exp \left\{ \frac{b_1(\lambda_i^p - \lambda_i^t)}{\lambda_i^p \lambda_i^t} \right\} \frac{1 + \operatorname{erf}\left(\frac{\lambda_i^t}{\sqrt{2}\sigma_0}\right)}{1 + \operatorname{erf}\left(\frac{\lambda_i^p}{\sqrt{2}\sigma_0}\right)}. \end{aligned} \quad (27)$$

In practice, parameters  $\sigma_0$  and  $\varepsilon$  are tuned during inference so that around 40% proposals are accepted, according to a heuristic rule in [35].

The MH sampling step for  $\alpha$  in Sampler 4 is designed in the same way. Proposals are drawn from the Gamma distribution:  $q(\alpha|\alpha^t) \propto \alpha^{a_2 - 1 + M_k} e^{-(1+b_2)\alpha}$ . Hence, the acceptance probability is:

$$A(\alpha^p|\alpha^t) = \min \left\{ 1, \frac{e^{-\alpha^t} \sum_{i=1}^{|\mathcal{M}|} (\alpha^t)^{M_i} (M_i!)^{-1}}{e^{-\alpha^p} \sum_{i=1}^{|\mathcal{M}|} (\alpha^p)^{M_i} (M_i!)^{-1}} \right\}. \quad (28)$$

Note that step 4 is independent on the other sampling steps. That is because hyperparameter  $\alpha$  is only related to  $\mathcal{M}_k$  that is pre-fixed in this section. As a result, step 4 can be removed from the sampler without affecting the convergence property. However, if  $\mathcal{M}_k$  needs to be sampled (unknown topology), step 4 must be maintained in the sampler.

### B. Sampler with unknown topology

If the network topology is unknown,  $\mathcal{M}_k$  is treated as a random variable and needs to be sampled for topology detection. To sample from (14), a blocked Gibbs sampler (Sampler 5) is applied as the last section. Since the dimension of  $W$ ,  $\beta$  and  $\lambda$  is dependent on  $\mathcal{M}_k$ , these random variables are grouped together.

---

#### Sampler 5: Blocked Gibbs sampler

---

- 1: Sample  $p(W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \mathcal{M}_k^{t+1} | \sigma^t, \alpha^t, Y)$
  - 2: Sample  $p(\sigma^{t+1} | W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \alpha^t, \mathcal{M}_k^{t+1}, Y)$
  - 3: Sample  $p(\alpha^{t+1} | W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \sigma^{t+1}, \mathcal{M}_k^{t+1}, Y)$
- 

Since the random variables of steps 1 and 3 in Sampler 5 cannot be sampled directly, these two steps are implemented using the MH method, leading to a MH-within-Gibbs sampler (Sampler 6).

---

#### Sampler 6: MH-within-Gibbs sampler

---

- 1: Sample  $p(W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \mathcal{M}_k^{t+1} | \sigma^t, \alpha^t, Y)$  using MH
  - 2: Sample  $p(\sigma^{t+1} | W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \alpha^t, \mathcal{M}_k^{t+1}, Y)$
  - 3: Sample  $p(\alpha^{t+1} | W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \sigma^{t+1}, \mathcal{M}_k^{t+1}, Y)$  using MH
- 

According to the rule of marginalization, after marginalizing  $W$  out from step 1, one must sample  $W$  immediately from their conditional distribution in the next step. The resulting MH-within-PCG sampler is shown in Sampler 7.

---

#### Sampler 7: MH-within-PCG sampler

---

- 1: Sample  $p(\beta^{t+1}, \lambda^{t+1}, \mathcal{M}_k^{t+1} | \sigma^t, \alpha^t, Y)$  using MH
  - 2: Sample  $p(W^{t+1} | \beta^{t+1}, \lambda^{t+1}, \sigma^t, \alpha^t, \mathcal{M}_k^{t+1}, Y)$
  - 3: Sample  $p(\sigma^{t+1} | W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \alpha^t, \mathcal{M}_k^{t+1}, Y)$
  - 4: Sample  $p(\alpha^{t+1} | W^{t+1}, \beta^{t+1}, \lambda^{t+1}, \sigma^{t+1}, \mathcal{M}_k^{t+1}, Y)$  using MH
- 

Since  $\mathcal{M}_k$  is fixed for steps 2, 3 and 4 in Sampler 7, these sampling steps can be implemented in the same way as Sampler 4. Sampler 7 explores different model structures (topologies) via step 1. Since the sampling space of step 1 is composed of multiple subspaces of different dimensionality, RJMCMC is applied to explore the distribution sufficiently.

To realize effective jumps between the parameter subspaces, three types of moves are designed for the Markov chain. These moves allow the Markov state to visit all subspaces.

**Birth Move:** The number of links in the next state,  $M_k^{t+1}$  is one more than that of the current state (i.e.  $M_k^{t+1} = M_k^t + 1$ ). Furthermore, the zero structure of  $\mathcal{M}_k^{t+1}$  and  $\mathcal{M}_k^t$  only differs at one entry. For example, the Boolean structure of  $\mathcal{M}_k^t$  is  $[1 \ 0 \ 0]$  and that of  $\mathcal{M}_k^{t+1}$  is  $[1 \ 0 \ 1]$ .

**Death Move:** The number of links in the next state,  $M_k^{t+1}$  is one less than that of the current state (i.e.  $M_k^{t+1} = M_k^t - 1$ ). Furthermore, the zero structure of  $\mathcal{M}_k^{t+1}$  and  $\mathcal{M}_k^t$  only differs at one entry. For example, the Boolean structure of  $\mathcal{M}_k^t$  is  $[1 \ 0 \ 1]$  and that of  $\mathcal{M}_k^{t+1}$  is  $[1 \ 0 \ 0]$ .

**Update Move:** The topology of the network is unchanged in the next state (i.e.  $M_k^{t+1} = M_k^t$ ) but the other random variables are updated.

The birth and death moves of RJMCMC encourage a global search of the parameter subspaces, leading to a thorough exploration of network topology. The death move is equivalent

to setting an ARD parameter  $\lambda_i$  to zero whilst the birth move reverses this process by retrieving non-zero  $\lambda_i$ . As a result, the effect of ARD is maximally activated to promote sparse topologies. The update move samples hyperparameters of the kernel functions that control system dynamics to interpret datasets.

Kernel based methods also apply ARD for topology detection, where the sparsity profile of ARD parameters determines network topology. However, due to local optimal solutions and numerical errors, the estimated ARD parameters are often not strictly zero. One can try different initial points to somehow avoid local optima or employ certain model selection strategies (e.g. backward selection) to enforce sparsity. However, since these schemes either pick local and global optima equally likely or implement algorithms repeatedly, they raise computational cost and can be very inefficient.

The main advantage of RJMCMC is that it infers networks effectively. RJMCMC detects network topology and identifies system dynamics in different ways whilst kernel based methods accomplish both tasks by hyperparameter estimation. RJMCMC infers network topology by visiting different parameter subspaces while system dynamics are estimated through drawing samples from these subspaces. Therefore, RJMCMC is able to avoid many local maxima of the target distribution. In addition, the decision of jumps is made based on the trade-off between data-fitting and sparsity penalties. Hence, RJMCMC visits topologies that are close to the ground truths more frequently, which reduces computational cost and improves inference accuracy.

To realize birth and death moves, the following algorithms (Algorithm 1 and 2) are proposed.

---

#### Algorithm 1 Birth Move

---

- 1: With probability  $P_B$ , choose Birth move.
- 2: Select node  $i$  to be added to the current topology based on the Uniform distribution:  $q_B(i | \mathcal{M}_k^t) = \frac{1}{M_1 - M_k^t}$ .
- 3: Draw proposals  $\beta_i^p$  and  $\lambda_i^p$  from the proposal distribution  $q_B(\beta_i, \lambda_i) = q_B(\beta_i)q_B(\lambda_i)$ , where

$$q_B(\lambda_i) = IG(\lambda_i; a_1, b_1)$$

$$TC/SS : q_B(\beta_i) = U(\beta_i; 0, 1) \quad (29)$$

$$DC : q_B(\beta_i) = U(\beta_{i1}; 0, 1)U(\beta_{i2}; -1, 1)$$

- 4: Insert  $\beta_i^p$  and  $\lambda_i^p$  into  $\beta^t$  and  $\lambda^t$ , respectively to generate  $\beta^{t+1}$  and  $\lambda^{t+1}$ . Change matrix  $\Phi$  accordingly.
  - 5: Accept with probability  $A_B$ .
- 

---

#### Algorithm 2 Death Move

---

- 1: With probability  $P_D$ , choose Death move.
  - 2: Select node  $i$  to be removed from the current topology based on the Uniform distribution:  $q_D(i | \mathcal{M}_k^t) = \frac{1}{M_k^t - 1}$  where the auto-regression terms are always kept.
  - 3: Remove  $\beta_i^t$  and  $\lambda_i^t$  from  $\beta^t$  and  $\lambda^t$  respectively with other components unchanged. Change matrix  $\Phi$  accordingly.
  - 4: Accept with probability  $A_D$ .
-

The acceptance probability for birth and death moves is calculated based on (21):

$$\begin{aligned} & A_B(\beta^p, \lambda^p, \mathcal{M}_k^p | \beta^t, \lambda^t, \mathcal{M}_k^t) \\ &= \min\{1, r_B(\beta^p, \lambda^p, \mathcal{M}_k^p | \beta^t, \lambda^t, \mathcal{M}_k^t)\}, \\ & A_D(\beta^p, \lambda^p, \mathcal{M}_k^p | \beta^t, \lambda^t, \mathcal{M}_k^t) \\ &= \min\{1, r_D(\beta^p, \lambda^p, \mathcal{M}_k^p | \beta^t, \lambda^t, \mathcal{M}_k^t)\}, \end{aligned} \quad (30)$$

where

$$\begin{aligned} & r_B(\beta^p, \lambda^p, \mathcal{M}_k^p | \beta^t, \lambda^t, \mathcal{M}_k^t) \\ &= \left[ \prod_{j=1}^L \frac{\exp\{-\frac{1}{2} Y_j' (\sigma_j^t I + \Phi_j \Lambda^p K^p \Phi_j')^{-1} Y_j\} |\sigma_j^t I + \Phi_j \Lambda^p K^p \Phi_j'|^{-\frac{1}{2}}}{\exp\{-\frac{1}{2} Y_j' (\sigma_j^t I + \Phi_j \Lambda^t K^t \Phi_j')^{-1} Y_j\} |\sigma_j^t I + \Phi_j \Lambda^t K^t \Phi_j'|^{-\frac{1}{2}}} \right] \\ & \times \frac{P_D \alpha^t (M_1 - M_k^t)}{P_B M_k^p (M_k^p - 1)}, \\ & r_D(\beta^p, \lambda^p, \mathcal{M}_k^p | \beta^t, \lambda^t, \mathcal{M}_k^t) = r_B^{-1}(\beta^t, \lambda^t, \mathcal{M}_k^t | \beta^p, \lambda^p, \mathcal{M}_k^p). \end{aligned} \quad (31)$$

Finally, the update move is shown in Algorithm 3. Since the topology is fixed, the proposal distributions and the acceptance probability are exactly the same with those of Sampler 4.

---

#### Algorithm 3 Update Move

---

- 1: With probability  $P_U$ , choose Update move.
  - 2: Draw proposals  $\beta^p$  and  $\lambda^p$  using (25) and (22), respectively.
  - 3: Accept with probability  $A_U$ .
- 

Note that the probability of three moves depends on  $\mathcal{M}_k^t$  as follows:

$$\begin{aligned} P_B &= \begin{cases} 0.3 & 1 < M_k^t < M_1 \\ 0 & M_k^t = M_1 \\ 0.6 & M_k^t = 1 \end{cases}, \\ P_D &= \begin{cases} 0.3 & 1 < M_k^t < M_1 \\ 0.6 & M_k^t = M_1 \\ 0 & M_k^t = 1 \end{cases}, \\ P_U &= 1 - P_B - P_D. \end{aligned} \quad (32)$$

To conclude, Algorithm 4 presents network inference using RJMCMC.

---

#### Algorithm 4 RJMCMC for network inference

---

- 1: Initialize  $W^0, \beta^0, \lambda^0, \sigma^0, \alpha^0, \mathcal{M}_k^0$ .
  - 2: **for**  $t = 1 : t_{max}$  **do**
  - 3:   Sample  $P_{move}$  from  $U(0, 1)$ .
  - 4:   **if**  $P_{move} \leq P_B$  **then**
  - 5:     Execute Birth Move (Algorithm (1)).
  - 6:   **else if**  $P_{move} \leq P_B + P_D$  **then**
  - 7:     Execute Death Move (Algorithm (2)).
  - 8:   **else**
  - 9:     Execute Update Move (Algorithm (3)).
  - 10:   **end if**
  - 11:   Sample  $W^t$  according to (16).
  - 12:   Sample  $\sigma^t$  according to (16).
  - 13:   Sample  $\alpha^t$  using step 4 in Sampler 4.
  - 14: **end for**
  - 15: Store  $\{W^t\}, \{\mathcal{M}_k^t\}$  and  $\{\sigma^t\}$ .
- 

#### C. Detection of topology and estimation of parameters

Detection of network topology is based on the marginal posterior distribution of model structures,  $p(\mathcal{M}_k | Y)$ . By the merit of RJMCMC, one can approximate the true distribution using the empirical distribution constructed from the samples:

$$P(\mathcal{M}_k = \mathcal{M}_i | Y) = \frac{1}{t_{max}} \sum_{t=1}^{t_{max}} \mathbf{1}_{\mathcal{M}_i}(\mathcal{M}_k^t), \quad (33)$$

where

$$\mathbf{1}_x(y) = \begin{cases} 1 & y = x \\ 0 & y \neq x \end{cases}. \quad (34)$$

Given the empirical distribution (33), the most likely network topology is estimated based on the maximum a posteriori method (MAP):  $\mathcal{M}_{opt} = \max_{\mathcal{M}_k} P(\mathcal{M}_k | Y)$ .

Often, one prefers to evaluate the confidence of all links. Under this circumstance, the generated topology is fully connected and the confidence of links is reflected by their probability. With the samples of network topology  $\{\mathcal{M}_k^t\}$ , the probability of the link from node  $j$  to node  $i$  ( $j \rightarrow i$ ) is estimated as follows:

$$\begin{aligned} P(j \rightarrow i | Y) &= \sum_{k=1}^{|\mathcal{M}|} P(j \rightarrow i, \mathcal{M}_k | Y) \\ &= \frac{1}{t_{max}} \sum_{\mathcal{M}_q | j \rightarrow i \in \mathcal{M}_q} \sum_{t=1}^{t_{max}} \mathbf{1}_{\mathcal{M}_q}(\mathcal{M}_k^t), \end{aligned} \quad (35)$$

where  $j \rightarrow i \in \mathcal{M}_q$  means that link  $j \rightarrow i$  is contained in topology  $\mathcal{M}_q$ .

Finally, impulse responses and noise variance are estimated for model simulation and prediction as follows:

$$\begin{aligned} \hat{W} &= E(W | \mathcal{M}_k = \mathcal{M}_{opt}, Y) = \frac{\sum_{t=1}^{t_{max}} \mathbf{1}_{\mathcal{M}_{opt}}(\mathcal{M}_k^t) W^t}{\sum_{t=1}^{t_{max}} \mathbf{1}_{\mathcal{M}_{opt}}(\mathcal{M}_k^t)}, \\ \hat{\sigma} &= E(\sigma | \mathcal{M}_k = \mathcal{M}_{opt}, Y) = \frac{\sum_{t=1}^{t_{max}} \mathbf{1}_{\mathcal{M}_{opt}}(\mathcal{M}_k^t) \sigma^t}{\sum_{t=1}^{t_{max}} \mathbf{1}_{\mathcal{M}_{opt}}(\mathcal{M}_k^t)}. \end{aligned} \quad (36)$$

## V. SIMULATIONS

We conducted two series of Monte Carlo simulations, and compared our method with the following state-of-the-art approaches: Kernel-based methods [14], iCheMA [1], GENIE3 [3] and dynGENIE3 [2]. The first set of simulations generated random DSF networks with different types of topologies (including a ring structure). They were simulated under various noise levels and inferred using time series data of various lengths. The second set of simulations investigated the algorithm performance when inferring real-world networks. In particular, it considered a well known synthetic gene regulatory network of the circadian clock of *Arabidopsis thaliana*.

Four criteria evaluated the performance of algorithms, namely true positive rate (TPR), precision (PREC), area under the receiver operating characteristic curve (AUROC) and area under the precision recall curve (AUPREC). TPR shows the percentage of the true links in the ground truths that are successfully inferred. Precision (PREC) equates to the rate

of the correct links over all the inferred links. TPR and PREC together indicate the accuracy of inferred networks. If TPR is low, the inferred network misses many true links, thus lacking of useful information; when PREC is low, the generated network is not reliable. Whilst TPR and PREC evaluate the most likely topologies, AUROC and AUPREC are more suitable for the methods that estimate the confidence of all links [1]. The receiver operating characteristic curve and the precision recall curve are generated by selecting links with an increasing threshold on confidence. AUROC and AUPREC reveal similar information of TPR and PREC, respectively.

To investigate the accuracy of estimated system dynamics, we conducted the one-step ahead prediction of the validation dataset using the identified models where the validation dataset was not used for inference. Outputs  $y(t)$  were predicted based on the past measurements before  $t-1$ . The prediction accuracy is measured based on the following metric:

$$fit = 1 - \frac{\|y - \hat{y}\|}{\|y - \bar{y}\|}. \quad (37)$$

where  $y$  are the validation data,  $\hat{y}$  are the predicted outputs and  $\bar{y}$  are the mean of the validation data. The average of the fitness of all nodes is calculated.

#### A. Random DSF networks

100 discrete-time networks were generated with random sparse topologies. These networks were described by linear state space models whose state variables represented nodes. All networks contained 15 nodes, among which 10 nodes were measured. Each measured node was independently driven by an input (known) and process noise (unknown). To generate linear state space models, system matrices  $A \in \mathcal{R}^{15 \times 15}$  were yielded randomly using the function  $sprandn(n, n, density)$  in Matlab. To guarantee system stability, only Hurwitz matrices  $A$  (i.e. matrices without eigenvalues outside of the unit circle) were kept for simulations. Only the first 10 states were measured for inference. These states were independently driven by an input and process noise. No isolated nodes existed in the network. Figure 1 displays one example of a simulated network.

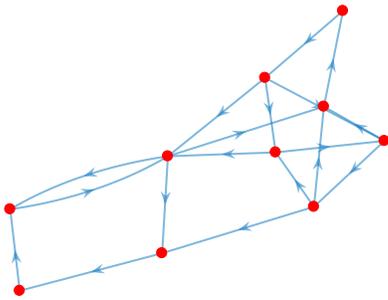


Fig. 1. The structure of a randomly generated sparse network of measured nodes. Solid lines with arrows represent links. Red circles denote nodes.

To simulate the models, inputs and process noise were both i.i.d. white Gaussian signals. The variance of inputs was fixed to 1 whilst that of process noise varied. The Signal-to-Noise

ratio is defined as  $SNR = 10 \log \frac{\sigma_u}{\sigma_e}$  where  $\sigma_u$  and  $\sigma_e$  are the signal variance of inputs and noise, respectively. Time series data were collected for inference with various lengths between 45 to 1000. The truncation length of impulse responses for model (5) was set to 20.

TPR and PREC are calculated given the most likely topologies. Their averaged values over 100 trials are recorded in Table I–III for different time series lengths. GENIE3 is not included in these tables since its TPR and PREC highly depend on the threshold tuned to select links. DynGENIE3 and iCheMA are not applied in this section since the simulated models are discrete-time systems. In the best-case scenario (no process noise), our method outperforms kernel based methods in all cases. RJMCMC endowed with difference kernel functions present similar results. In particular, given sufficiently long time series ( $\geq 85$ ), our method offers perfect inference. In contrast, Kernel\_TC presents the weakest result. TPR of Kernel\_DC stays below 90%.

TABLE I  
INFERENCE OF RANDOM NETWORKS WITH NO NOISE

	No Noise					
	45		65		85	
	PREC	TPR	PREC	TPR	PREC	TPR
Kernel_DC	91.2	54.7	95.7	73.5	99.4	84.2
Kernel_SS	82.8	60.7	89.6	93.2	99.9	99.9
Kernel_TC	50.1	17.0	76.9	29.4	91.2	40.5
RJMCMC_DC	99.4	90.5	100	99.3	100	100
RJMCMC_SS	93.4	91.2	100	99.3	100	100
RJMCMC_TC	99.6	91.6	100	99.3	100	100

TABLE II  
INFERENCE OF RANDOM NETWORKS WITH NOISE THAT HAS 10dB SNR

	10dB					
	100		200		300	
	PREC	TPR	PREC	TPR	PREC	TPR
Kernel_DC	91.9	75.5	97.2	84.0	98.0	87.1
Kernel_SS	74.7	82.9	80.7	88.8	87.8	89.0
Kernel_TC	87.3	47.5	99.6	67.6	100	74.2
RJMCMC_DC	97.0	80.0	98.2	87.4	98.5	89.4
RJMCMC_SS	68.3	85.8	80.7	90.1	82.4	93.2
RJMCMC_TC	98.1	81.6	98.0	88.5	98.7	90.1

TABLE III  
INFERENCE OF RANDOM NETWORKS WITH PURE NOISE

	No Input/Pure noise					
	300		500		1000	
	PREC	TPR	PREC	TPR	PREC	TPR
Kernel_DC	81.0	71.5	85.6	73.9	97.2	75.3
Kernel_SS	66.4	68.8	80.5	70.9	82.0	72.8
Kernel_TC	81.6	59.5	89.2	66.2	93.4	71.3
RJMCMC_DC	96.2	69.2	98.8	76.3	97.5	83
RJMCMC_SS	78.6	74.3	87.1	76.5	88.4	84.5
RJMCMC_TC	95.7	70.9	96.7	76.7	98.9	81.8

As  $SNR$  decreases to 10dB, our method exhibits different performances with distinct kernel functions. RJMCMC\_DC and RJMCMC\_TC are both superior to RJMCMC\_SS, and show similar performance. In particular, RJMCMC\_DC and RJMCMC\_TC are always capable of producing reliable networks ( $PREC \geq 97\%$ ). As the number of data points

increases, they successfully capture most true links ( $TPR \approx 90\%$ ). As with previous simulations, kernel based methods have poorer performance than RJMCMC.

With only process noise driving (no inputs), the performance improvement of our method is clearly evident compared with kernel based methods. It is remarkable that PREC of RJMCMC\_DC and RJMCMC\_TC is always above 95%, indicating the inferred networks are highly reliable.

Next, we calculate AUROC and AUPREC to allow a comparison with GENIE3. For kernel based methods, we resorted to [36] to calculate the confidence of all links as  $P(j \rightarrow i|Y) = \frac{\|w_{ij}\|}{\|W\|}$ . Simulations indicate that our method consistently outperforms kernel based methods and GENIE3 in all cases. Due to the lack of space, only the result of  $10dB$  SNR are present in Table IV. Table IV shows that AUROC and AUPREC of our method always stay above 90% whilst those of GENIE3 are consistently below 85%. In addition, our method is still superior to kernel based methods.

TABLE IV  
INFERENCE OF RANDOM NETWORKS WITH NOISE THAT HAS  $10dB$  SNR

	10dB					
	100		200		300	
	AUPREC	AUROC	AUPREC	AUROC	AUPREC	AUROC
GENIE3	68.2%	77.4%	73.0%	83.5%	74.1%	84.1%
Kernel_DC	89.7%	87.9%	93.6%	92.2%	94.5%	93.2%
Kernel_SS	88.2%	88.5%	94.4%	93.9%	95.6%	95.2%
Kernel_TC	75.2%	72.1%	87.6%	84.4%	90.8%	88.6%
RJMCMC_DC	90.3%	93.4%	95.1%	96.7%	96.2%	97.4%
RJMCMC_SS	90.8%	94.1%	95.2%	96.7%	96.3%	98.0%
RJMCMC_TC	92.3%	95.1%	94.8%	95.9%	97.2%	98.2%

The validation result is shown by the box plot in Figure 2. GENIE3 is not included since it cannot be used for model prediction. Our method has considerable better performance than kernel based methods with no noise. The prediction accuracy of RJMCMC\_TC and RJMCMC\_DC is better than kernel based methods for  $SNR = 10dB$ , especially with a low number of data points. Kernel\_TC always presents the weakest result. TPR of Kernel\_TC is much lower than PREC, which means Kernel\_TC achieves reliable inference at the cost of missing many true links. As a result, the estimated models of Kernel\_TC are not able to predict new datasets accurately.

### B. Ring networks

Next, we generated 100 discrete-time networks with a fixed ring structure (Figure 3) and then simulated following the same protocol of random DSFs. Each measured node was driven by independent process noise. Only one input entered the network through a single measured node. Since ring networks form closed feedback loops and are extremely sparse, they can be challenging to infer.

Table V presents the inference results given the most likely topologies. Simulations indicate that our method is superior to kernel based methods. In particular, RJMCMC\_DC and RJMCMC\_TC present the best results. PREC of these two cases always exceeds 90% and increases to 98% given 400 data points, with only 1 true link missed. For kernel based methods, either PREC or TPR is lower than the corresponding RJMCMC cases. Table VI compares our method with GENIE3. Similar to random DSFs, our method consistently

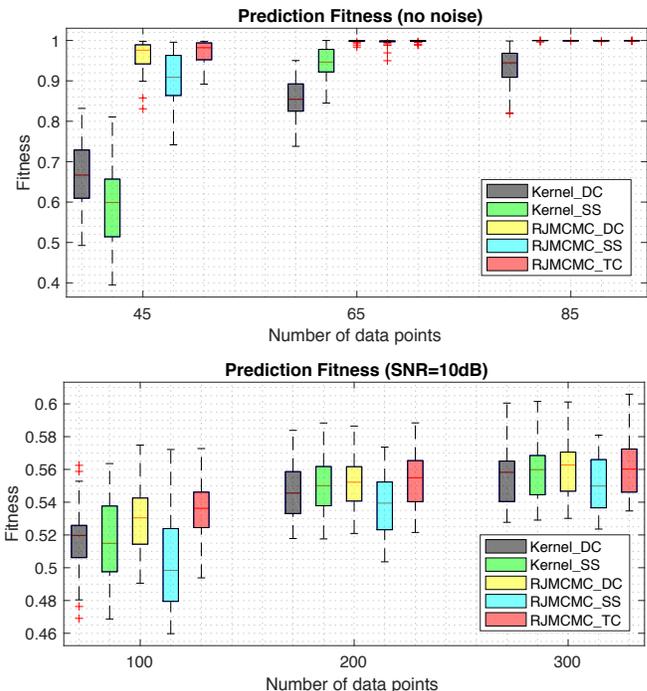


Fig. 2. Performance for randomly generated networks. x-axis denotes the number of data points. y-axis presents the scaled prediction fitness which is equal to  $10^{fit-1}$ . Since the prediction fitness of Kernel\_TC is in general much lower than the others, it is omitted in the figure. RJMCMC\_TC and RJMCMC\_DC are much better than kernel based methods with a low number data points.

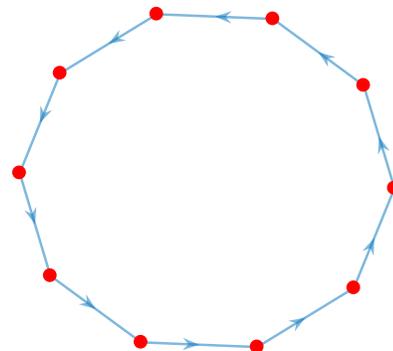


Fig. 3. A network with the ring structure.

outperforms kernel based methods and GENIE3. AUPREC of GENIE3 is below 70% in all cases, meaning the inferred networks are not as reliable.

TABLE V  
INFERENCE OF RING NETWORKS WITH  $10dB$  SNR

	10dB							
	100		200		300		400	
	PREC	TPR	PREC	TPR	PREC	TPR	PREC	TPR
Kernel_DC	54.4	77.0	75.8	82.3	85.0	85.5	90.5	86.5
Kernel_SS	41.2	78.5	62.8	83.5	71.8	90.0	71.3	89.0
Kernel_TC	76.9	19.3	98.4	39.3	96.8	56.0	97.6	67.8
RJMCMC_DC	92.0	67.3	94.5	79.3	94.9	85.5	98.1	88.8
RJMCMC_SS	60.7	70.0	81.1	80.3	81.2	85.5	85.9	88.3
RJMCMC_TC	91.2	66.3	95.3	80.8	96.6	86.8	98.0	89.3

The validation results in Figure 4 show that our method

TABLE VI  
INFERENCE OF RING NETWORKS WITH 10dB SNR

	10dB							
	100		200		300		400	
	AUPREC	AUROC	AUPREC	AUROC	AUPREC	AUROC	AUPREC	AUROC
GENIE3	62.4%	74.7%	66.7%	77.2%	67.9%	78.3%	67.6%	78.0%
Kernel_DC	70.3%	85.6%	85.6%	90.4%	91.2%	92.7%	92.6%	94.3%
Kernel_SS	71.8%	84.9%	87.6%	92.0%	90.8%	92.3%	92.1%	93.2%
Kernel_TC	57.5%	56.9%	71.7%	69.0%	78.9%	76.5%	85.9%	84.5%
RJMCMC_DC	76.8%	85.8%	87.6%	91.2%	91.4%	94.0%	92.6%	94.7%
RJMCMC_SS	76.2%	84.6%	85.3%	89.0%	90.4%	92.7%	91.8%	93.8%
RJMCMC_TC	80.0%	87.6%	87.6%	92.5%	92.2%	94.3%	92.6%	95.0%

outperforms kernel based methods, especially with a low number of data points.

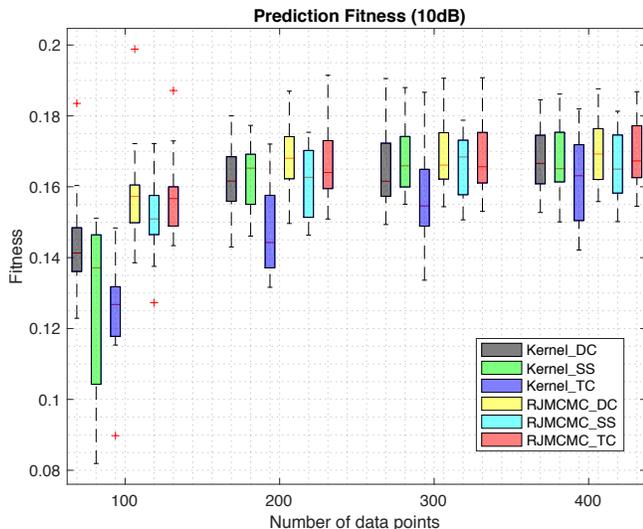


Fig. 4. Prediction of ring networks. x-axis denotes the number of data points. y-axis presents the scaled prediction fitness which is equal to  $10^{fit-1}$ . It is clearly evident that our approach outperforms kernel based methods, especially with a low number data points.

### C. Synthetic circadian clock network

In above two simulations, the ground truth models fall exactly in the proposed model class. However, most real-world networks are nonlinear. Under our framework, linear models are used as approximations to nonlinear dynamics. The effectiveness of our method was tested on a synthetic model of the circadian clock (Millar 10 [37]). Since this model is based on ODEs, iCheMA and dynGENIE3 were considered.

Millar 10 describes a circadian clock consisting of 7 genes along with their associated proteins, which amounts to 19 nodes in total. The system is driven by light signals. The detailed mathematical model can be found in [37]. Simulations aim to produce synthetic microarray data similar to those obtained in real life experiments. The time window for data collection was 44 hours, with a sampling frequency of 1 hour. Hence, only 44 data points were obtained from each simulation. Most importantly, protein data were not available for inference, which is typical in genome-wide experiments. Therefore, the network was inferred at the transcriptional level, describing the connectivity among 7 clock genes. The model was simulated for four days on light-dark (LD) cycles followed by three days of constant light (LL). The simulations were

repeated 50 times, one of which is shown in Figure 5. To avoid the transition due to the initial condition, the simulated data of the first two days were discarded. Time windows of LDLD (0h-44h), LDLL (24h-68h), LLLL (48h-92h) and steady state (72h-116h) were adopted for data collection. Considering that only 44 data points were available for inference, the length of truncated impulse responses was set to 10.

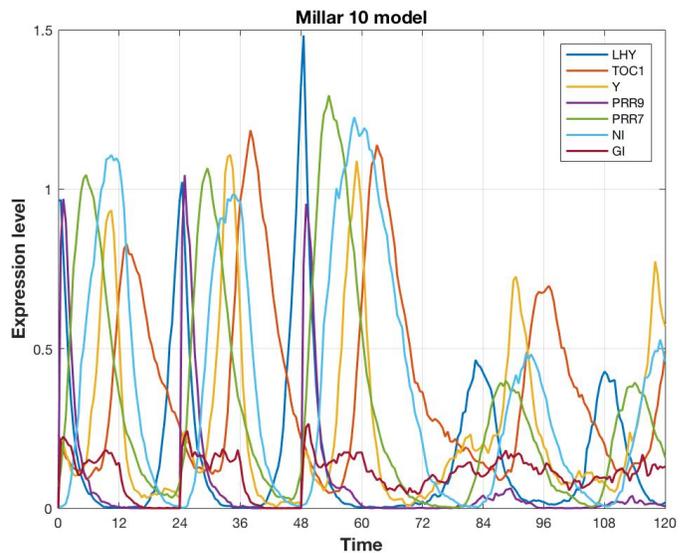


Fig. 5. Millar 10 network with 7 genes.

The inference results are presented in Table VII. RJMCMC\_SS outperforms all the other methods in most cases. In particular, under time window LDLL, both AUPREC and AUROC of RJMCMC\_SS are above 70%. These inference results indicate that our method is able to infer complex system dynamics. The inference accuracy of our method is markedly improved compared with kernel based methods. In fact, kernel based methods perform poorly with this model, as the inferred networks are unreliable with most true links missing. In most cases, iCheMA and dynGENIE3 have poorer performance than our method. Especially in LDLL, iCheMA and dynGENIE3 seem to fail to capture the added information contained on the transient data (due to the light transition from LD to LL).

Simulations imply that our method is reliable when dealing with real-world networks, especially for the cases where full state measurements are unavailable. Therefore, our method can be applied under a wide range of contexts such as biological networks, power grids and communication systems.

## VI. CONCLUSION

This paper applies Gaussian processes and RJMCMC to infer discrete-time linear, sparse networks. DSF models are used to describe the target network with unmeasured nodes encoded via transfer functions. The models are expressed in a non-parametric way to avoid estimating model complexity. Gaussian processes are used to impose stable impulse responses. To sufficiently explore the full Bayesian model, RJMCMC is applied to draw samples from the space that is composed of subspaces of different dimensionality. By traversing parameter subspaces, RJMCMC greatly improves

TABLE VII  
INFERENCE RESULTS OF THE CIRCADIAN CLOCK MODEL

	LDLD		LDLL		LLLL		Steady State	
	AUPREC	AUPROC	AUPREC	AUROC	AUPREC	AUROC	AUPREC	AUROC
iCheMA	62.3%	66.4%	64.2%	65.4%	66.8%	69.4%	56.4%	64.7%
dynGENIE3	58.8%	66.5%	56.1%	64.5%	59.1%	63.4%	61.5%	70.9%
Kernel_DC	45.4%	54.9%	55.3%	63.1%	39.4%	53.6%	35.0%	48.9%
Kernel_SS	38.8%	51.3%	51.6%	63.3%	39.7%	54.2%	37.4%	51.7%
Kernel_TC	36.5%	48.6%	43.7%	55.5%	38.9%	51.7%	35.5%	47.4%
RJMCMC_DC	61.1%	64.8%	66.7%	68.7%	57.2%	61.8%	54.0%	58.2%
RJMCMC_SS	63.8%	69.4%	73.4%	76.5%	62.7%	66.3%	62.8%	64.6%
RJMCMC_TC	61.8%	68.0%	68.9%	72.2%	57.0%	61.2%	54.5%	59.1%

the inference accuracy. In addition, our method can produce the most likely topology and estimate confidence of all links. Monte Carlo simulations demonstrate our method superior to the state-of-the-art methods, including kernel based methods, iCheMA, GENIE3 and dynGENIE3.

Overall, the value of this approach are in generating reliable inference results, and being robust to experimental conditions, including the number of data points, types of topologies and noise levels. Our method is applicable to a wide range of real-world networks, where full state measurements are not available. According to simulations, our method can also be used to study continuous-time nonlinear models, such as the  $Ca^{2+}$  signalling network of *Arabidopsis*.

The method does, however, have some limitations. Since the method relies on numerical sampling, the computational cost is heavy when dealing with large-scale networks. Moreover, the method for identification of continuous-time systems requires relatively high sampling frequency and equal sampling steps. Finally, additional work is required to extend the method to capture nonlinearities and continuous-time networks.

## REFERENCES

- [1] A. Aderhold, D. Husmeier, and M. Grzegorzczak, "Approximate bayesian inference in semi-mechanistic models," *Statistics and Computing*, vol. 27, p. 1003, 2017.
- [2] V. A. Huynh-Thu and P. Geurts, "DynGENIE3: Dynamical GENIE3 for the inference of gene networks from time series expression data," *Scientific Reports*, vol. 8, no. 1, pp. 1–12, 2018.
- [3] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts, "Inferring regulatory networks from expression data using tree-based methods," *PLoS ONE*, vol. 5, no. 9, pp. 1–10, 2010.
- [4] A. Greenfield, A. Madar, H. Ostrer, and R. Bonneau, "DREAM4: Combining genetic and dynamic information to identify biological networks and Dynamical Models," *PLoS ONE*, vol. 5, no. 10, 2010.
- [5] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. The MIT Press, 2006.
- [6] G. Wahba, *Spline Models for Observational Data*. SIAM: Society for Industrial and Applied Mathematics, 1990.
- [7] T. Chen, M. Andersen, L. Ljung, A. Chiuso, and G. Pillonetto, "System identification via sparse multiple kernel-based regularization using sequential convex optimization techniques," *Automatica*, vol. 59(11), pp. 1–33, 2014.
- [8] 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. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0005109816000376>
- [9] G. Pillonetto, M. H. Quang, and A. Chiuso, "A new kernel-based approach for nonlinear system identification," *IEEE Transactions on Automatic Control*, vol. 56, no. 12, pp. 2825–2840, Dec 2011.
- [10] G. Pillonetto, F. Dinuzzo, T. Chen, G. D. Nicolao, and L. Ljung, "Kernel methods in system identification, machine learning and function estimation: A survey," *Automatica*, vol. 50(3), pp. 657–682, 2014.
- [11] G. Pillonetto and G. D. Nicolao, "A new kernel-based approach for linear system identification," *Automatica*, vol. 46(1), pp. 81–93, 2010.
- [12] C. Bishop., *Pattern recognition and machine learning*. Springer New York, 2006.
- [13] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
- [14] A. Chiuso and G. Pillonetto, "A Bayesian approach to sparse dynamic network identification," *Automatica*, pp. 1553–1565, 2012.
- [15] T. Chen, H. Ohlsson, and L. Ljung, "On the estimation of transfer functions, regularizations and gaussian processes-revisited," *Automatica*, vol. 48, no. 8, pp. 1525 – 1535, 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0005109812001999>
- [16] G. Pillonetto, A. Chiuso, and G. Nicolao, "Prediction error identification of linear systems: A nonparametric gaussian regression approach," *Automatica*, vol. 47, no. 2, pp. 291 – 305, 2011. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0005109810004875>
- [17] P. Green, "Reversible jump markov monte carlo computation and bayesian model determination," *Biometrika*, vol. 82, pp. 711–732, 1995.
- [18] S. Brooks and B. Morgan, "Optimization using simulated annealing," *Journal of the Royal Statistical Society*, vol. 44, pp. 241–257, 1995.
- [19] S. P. Brooks, N. Friel, and R. King, "Classical model selection via simulated annealing," *Journal of the Royal Statistical Society*, vol. 65, pp. 503–520, 2003.
- [20] C. Andrieu, N. D. Freitas, and A. Doucet, "Reversible jump mcmc simulated annealing for neural networks,," in *Uncertainty in Artificial Intelligence Proceedings*, 2000, pp. 11–18.
- [21] C. Andrieu, N. de Freitas, and A. Douc, "Robust full bayesian learning for radial basis networks," *Neural computation*, vol. 13, pp. 2359–2407, 2001.
- [22] C. Andrieu and A. Doucet, "Joint bayesian model selection and estimation of noisy sinusoids via reversible jump mcmc," *IEEE Transactions on Signal Processing*, vol. 47, pp. 2667–2676, 1999.
- [23] T. Baldacchino, S. R. Anderson, and V. Kadiramanathan, "Computational system identification for bayesian narmax modelling," *Automatica*, vol. 49, pp. 2641–2651, 2013.
- [24] J. Vermaak, C. Andrieu, A. Doucet, and S. J. Godsill, "Reversible jump markov chain monte carlo strategies for bayesian model selection in autoregressive processes," *Journal of Time Series Analysis*, vol. 25, pp. 785–809, 2004.
- [25] J. Gonçalves and S. Warnick, "Necessary and sufficient conditions for dynamical structure reconstruction of lti networks," *IEEE Trans. Autom. Control*, vol. 53(7), pp. 1670–1674, 2008.
- [26] Y. Yuan, A. Rai, E. Yeung, G. Stan, S. Warnick, and J. Gonçalves, "A minimal realization technique for the dynamical structure function of a class of lti systems," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 2, pp. 301–311, June 2017.
- [27] Y. Yuan, G. Stan, S. Warnick, and J. Gonçalves, "Robust dynamical network structure reconstruction," *Automatica*, vol. 47, pp. 1230–1235, 2011.
- [28] F. Dinuzzo, "Kernels for linear time invariant system identification," *SIAM Journal on Control and Optimization*, vol. 53, no. 5, pp. 3299–3317, 2015. [Online]. Available: <https://doi.org/10.1137/130920319>
- [29] G. O. Roberts and S. K. Sahu, "Updating schemes, correlation structure, blocking and parameterization for the gibbs sampler," *Journal of the Royal Statistical Society*, vol. 59, pp. 291–317, 1997.
- [30] A. A. Johnson, G. L. Jones, and R. C. Neath, "Component-wise markov chain monte carlo: Uniform and geometric ergodicity under mixing and composition," *Statistical Science*, vol. 28, pp. 360–375, 2013.
- [31] D. A. van Dyk and X. Jiao, "Metropolis-hastings within partially collapsed gibbs samplers," *Journal of Computational and Graphical Statistics*, vol. 24, pp. 301–327, 2015.

- [32] D. A. van Dyk and T. Park, "Partially collapsed gibbs samplers: Theory and methods," *Journal of the American Statistical Association*, vol. 103, pp. 790–796, 2008.
- [33] D. I. Hastie, "Towards automatic reversible jump markov chain monte carlo," Ph.D. dissertation, University of Bristol, 2005.
- [34] R. Waagepetersen and D. Sorensen, "A tutorial on reversible jump mcmc with a view toward applications in qtl-mapping," *International Statistical Review*, vol. 69, no. 1, pp. 49–61, 2007. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1751-5823.2001.tb00479.x>
- [35] C. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer, 2004.
- [36] A. Aderhold, D. Husmeier, and M. Grzegorzczuk, "Statistical inference of regulatory networks for circadian regulation," *Statistical Applications in Genetics and Molecular Biology*, vol. 13, pp. 227–273, 2014.
- [37] A. Pokhilko, S. Hodge, K. Stratford, K. Knox, K. Edwards, A. Thomson, T. Mizuno, and A. Millar, "Data assimilation constrains new connections and components in a complex, eukaryotic circadian clock model." *Molecular systems biology*, vol. 6, pp. 1–10, 2010.