Fifty-Fifth Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
October 3-6, 2017

Reconstruction of Gene Regulatory Networks using an Error Filtering Learning Scheme

Ioannis Tzortzis, Christoforos N. Hadjicostis, and Laurent Mombaerts

***Abstract*— One of the fundamental and most challenging problems in system biology is the reconstruction of gene regulatory networks from input-output data based on nonlinear differential equations. This paper presents an approach to estimate the unknown nonlinearities and to identify the true network that generated the data, based on an error filtering learning scheme and a Lyapunov synthesis method. Unknown nonlinearities are modelled by networks using radial basis functions and model validation is performed by taking advantage of the so-called persistency of excitation of input signals, a condition that is shown to play a significant role in the problem of uncovering the true network structure. The proposed methodology and the theoretical results are validated through an illustrative example.**

I. Introduction

One of the fundamental and most challenging problems in systems biology is the reconstruction of a gene regulatory network from input-output data, in an effort to understand the interplay between different genes and consequently to explain the observed behavior of a particular biological system. The difficulty in adequately reconstructing a gene regulatory network from input-output data, arises mainly due to the nonlinear behavior of most biological systems, and due to limited information and noisy measurements.

Over the years many mathematical approaches have been proposed and applied to infer network structures from measurements. Good reviews for the developed approaches, which include Bayesian inference, information theory, Boolean networks, and ordinary differential equations, can be found, for example, in [1]–[3]. Extensive research has been performed to infer gene regulatory networks using linear time-invariant (LTI) systems [4]–[7], due to the existence of powerful analytic tools and the small number of embedded parameters. Necessary and sufficient conditions for network reconstruction of LTI systems are also provided in [8]. Even though mathematical methods based on LTI systems provide a tractable and useful option, these methods cannot guarantee recovery of the true nonlinear dynamical network.

In this paper we consider the realistic scenario that biological systems are typically nonlinear and we solve the problem of network reconstruction using an error filtering learning scheme and a Lyapunov synthesis method, adopted from adaptive approximation and nonlinear model identification frameworks. In particular, a learning model is designed and used to approximate the unknown nonlinearities by employing radial basis functions networks (i.e., appropriate weighted combinations of radial basis functions). The network weights are derived by utilizing a Lyapunov function, described by some stability properties, and updated based on an "error" training signal. In addition, and in contrast to the existing literature, this paper performs model validation based on the condition of persistency of excitation of the input signals, which plays an important role in the problem of network reconstruction.

Regarding the derivation of adaptive laws, a significant number of results have been developed by exploiting well-known optimization approaches such as the gradient (or steepest descent) method and the recursive least square method [9], [10]. Optimization approaches are mostly used for offline parameter estimation; in other words, all the data in a given time interval is gathered before it is processed to fit a model. The emphasis in this paper is to develop a novel approach for biological network reconstruction, suited to time series data and capable of simulating and identifying system changes that occur in response to biological stimulation.

The proposed approach combines dynamic modeling using differential equations with an error filtering scheme, in an effort to identify a predictive dynamical model. Despite the fact that many approaches are focused on simply inferring the topology of the network (who influences who) rather than the dynamical model itself [11], [12], (the latter obviously is a much harder task), the proposed approach could also prove valuable in settings where small gene networks need to be associated with predictive dynamical models. This requirement is particularly true in synthetic biology where small gene regulatory networks with strong nonlinearities are engineered to exhibit specific functions. The results of this paper indicate that the proposed approach may accurately enough infer gene regulatory networks from input-output data.

The remainder of the paper is organized as follows. In Section II we introduce the problem formulation and the proposed methodology, including an error filtering learning scheme and a Lyapunov synthesis method. In Section III the theoretical results are applied to a biologically inspired application by introducing a typical gene regulatory network and by working out an illustrative example which validates

This work was supported in part by the Cyprus Research Promotion Foundation (CRPF) under project KOINA/ERASySAPP/ERA.NET/1113/01. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of CRPF.

I. Tzortzis and C. N. Hadjicostis are with the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus. E-mails: {tzortzis.ioannis, chadjic}@ucy.ac.cy.

L. Mombaerts is with the Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Luxembourg City, Luxembourg. E-mail: laurent.mombaerts@uni.lu.

the results of the paper. Finally, in Section IV we conclude the paper with some summarizing remarks and possible future directions.

II. Problem Formulation and Methodology

Consider a nonlinear system of the form

$$\dot{y}(t) = f^*(y(t), u(t)), \tag{1}$$

where $u(t) \in \mathbb{R}^m$ is the input vector, $y(t) \in \mathbb{R}^n$ is the output vector and $f^* : \mathbb{R}^n \times \mathbb{R}^m \longmapsto \mathbb{R}^n$ is an unknown nonlinear function representing the system dynamics. Assume that f^* can be decomposed into two components, f and g, where f is function of $y(t)$ and g is a function of $u(t)$. Then, the nonlinear system (1) is expressed as

$$\dot{y}(t) = f(y(t)) + g(u(t)), \tag{2}$$

where f, g are unknown nonlinear functions to be approximated. To make the problem tractable one method is to replace the unknown functions f, g by some approximator functions of known structure and tunable parameters θ. Let $\widehat{f}(y;\theta_f^*)$ and $\widehat{g}(u;\theta_g^*)$ denote respectively the "best" approximations for f and g, where $\theta_f^* \in \mathbb{R}^{n_1}$, $\theta_g^* \in \mathbb{R}^{n_2}$ denote the optimal parameter vectors that minimize the maximum difference between the unknown functions and the approximator functions, i.e.,

$$\theta_f^* = \arg\min_{\theta_f \in \mathbb{R}^{n_1}} \Big\{ \sup_{y \in \mathcal{Y}} \big| f(y) - \widehat{f}(y;\theta_f^*) \big| \Big\}, \ \mathcal{Y} \subset \mathbb{R}^n,$$

$$\theta_g^* = \arg\min_{\theta_g \in \mathbb{R}^{n_2}} \Big\{ \sup_{u \in \mathcal{U}} \big| g(u) - \widehat{g}(u;\theta_g^*) \big| \Big\}, \ \mathcal{U} \subset \mathbb{R}^m.$$

Therefore, the nonlinear system (2) can be rewritten as

$$\dot{y}(t) = \widehat{f}(y(t);\theta_f^*) + \widehat{g}(u(t);\theta_g^*) + \delta(t), \tag{3}$$

with $\delta(t)$ denoting the approximation error after the best fit has been achieved and is given by

$$\delta(t) = f(y(t)) - \widehat{f}(y(t);\theta_f^*) + g(u(t)) - \widehat{g}(u(t);\theta_g^*).$$

A. RBF network models

The network models employed for capturing the unknown functions f and g are radial basis functions (RBF), and the approximator functions $\widehat{f}$ and $\widehat{g}$ in (3) are assumed to be decomposed to

$$\widehat{f}(y;\Theta_f^*) = \Theta_f^* \phi(y), \quad \text{and} \quad \widehat{g}(u;\Theta_g^*) = \Theta_g^* \phi(u), \tag{4a}$$

where $\Theta_f^* \in \mathbb{R}^{n \times n_1}$, $\Theta_g^* \in \mathbb{R}^{n \times n_2}$ denote the optimal parameter matrices, and the basis functions $\phi(y) \in \mathbb{R}^{n_1}$, $\phi(u) \in \mathbb{R}^{n_2}$ are of Gaussian form, defined element-wise as

$$\phi_i(y) = \exp\Big(-|y - c_{f,i}|^2 / \sigma_{f,i}^2 \Big), \ i = 1, 2, \ldots, n_1, \tag{5a}$$

$$\phi_j(u) = \exp\Big(-|u - c_{g,j}|^2 / \sigma_{g,j}^2 \Big), \ j = 1, 2, \ldots, n_2, \tag{5b}$$

where the widths $\sigma_{f,i}$, $\sigma_{g,j}$ and the centers $c_{f,i}$, $c_{g,j}$, $i = 1, 2, \ldots n_1$, $j = 1, 2, \ldots, n_2$, are chosen *a priori*, with the centers distributed uniformly in the region of interest. Hence, the only adjustable weights are Θ_f, Θ_g, which appear linearly with respect to the nonlinearities $\phi(y)$ and $\phi(u)$ respectively. Other types of candidate basis functions, which can be used to model the unknown functions f, g, include for example, various types of polynomials, sigmoidal functions, and Hill-type functions. Note that the emphasis of the methodology developed in this paper is on unknown nonlinearities that need to be estimated by nonlinear approximators, where the parameter estimates appear linearly with respect to the basis functions[1].

Assumption 2.1: The optimal parameter values Θ_f^* and Θ_g^* are bounded, so that $||\Theta_f^*||_F \leq M_f$ and $||\Theta_g^*||_F \leq M_g$ respectively, where M_f and M_g are known bounds. The norm $||\cdot||_F$ denotes the Frobenius norm [13], i.e., $||\Theta^*||_F = \mathrm{Tr}\{\Theta^* \Theta^{*T}\}$ where $\mathrm{Tr}\{\cdot\}$ denotes the trace of a matrix.

Substituting (4) in (3) we get

$$\dot{y}(t) = \Theta_f^* \phi(y(t)) + \Theta_g^* \phi(u(t)) + \delta(t), \tag{6}$$

with the approximation error

$$\delta(t) = f(y(t)) - \Theta_f^* \phi(y(t)) + g(u(t)) - \Theta_g^* \phi(u(t)).$$

Next, we design a learning model which will generate the output estimation error $e(t)$ (training signal) and will be used to approximate the unknown nonlinearities in the system. Then, we apply the Lyapunov synthesis method to derive adaptive laws for updating the parameter estimates of Θ_f^* and Θ_g^*.

B. Learning scheme and Lyapunov synthesis method

The results of this section follow from [14]. Consider Equation (6), which can be rewritten in the form

$$\dot{y}(t) = \lambda y(t) - \lambda y(t) + \Theta_f^* \phi(y(t)) + \Theta_g^* \phi(u(t)) + \delta(t), \tag{7}$$

where $\lambda > 0$ is a design constant. Based on (7), the learning model is described by

$$\dot{\widehat{y}}(t) = \lambda y(t) - \lambda \widehat{y}(t) + \widehat{\Theta}_f(t) \phi(y(t)) + \widehat{\Theta}_g(t) \phi(u(t)), \tag{8}$$

where the unknown "optimal" parameters Θ_f^* and Θ_g^* are replaced by their parameter estimates $\widehat{\Theta}_f(t)$ and $\widehat{\Theta}_g(t)$ respectively, and $\widehat{y}$ is the output of the learning model. Define the output estimation error by $e(t) \triangleq \widehat{y}(t) - y(t)$, and the parameter estimation errors by $\widetilde{\Theta}_f(t) \triangleq \widehat{\Theta}_f(t) - \Theta_f^*$ and $\widetilde{\Theta}_g(t) \triangleq \widehat{\Theta}_g(t) - \Theta_g^*$. Then, from (7) and (8) the output estimation error is expressed by

$$\dot{e}(t) = -\lambda e(t) + \widetilde{\Theta}_f(t) \phi(y(t)) + \widetilde{\Theta}_g(t) \phi(u(t)) - \delta(t). \tag{9}$$

The next step is to choose a suitable Lyapunov function candidate to derive adaptive laws for adjusting the parameter estimates. Toward this end, choose a Lyapunov function which is quadratic in the output estimation error e and the parameter's estimation error $\widetilde{\Theta}_f$ and $\widetilde{\Theta}_g$, e.g.,

$$V(e, \widetilde{\Theta}_f, \widetilde{\Theta}_g) = \frac{1}{2} e^T e + \frac{1}{2} \mathrm{Tr}\Big\{ \widetilde{\Theta}_f \Gamma_f^{-1} \widetilde{\Theta}_f^T \Big\} + \frac{1}{2} \mathrm{Tr}\Big\{ \widetilde{\Theta}_g \Gamma_f^{-1} \widetilde{\Theta}_g^T \Big\}, \tag{10}$$

[1]This should not be confused with nonlinear systems in which the nonlinearities are known but are multiplied with unknown parameters.

where $\Gamma_f \in \mathbb{R}^{n_1 \times n_1}$ and $\Gamma_g \in \mathbb{R}^{n_2 \times n_2}$ are positive-definite symmetric design matrices that will appear in the adaptive law of $\widehat{\Theta}_f$ and $\widehat{\Theta}_g$ respectively. Using (9), the time derivative of V in (10) is expressed as

$$\begin{aligned}\dot{V} =& -\lambda e^T e + \phi(y)^T \widetilde{\Theta}_f^T e + \phi(u)^T \widetilde{\Theta}_g^T e \\ &+ \mathrm{Tr}\left\{\dot{\widetilde{\Theta}}_f \Gamma_f^{-1} \widetilde{\Theta}_f^T\right\} + \mathrm{Tr}\left\{\dot{\widetilde{\Theta}}_g \Gamma_g^{-1} \widetilde{\Theta}_g^T\right\} - e^T\delta \\ \overset{(a)}{=}& -\lambda e^T e + \mathrm{Tr}\left\{e\phi(y)^T \widetilde{\Theta}_f^T + \dot{\widetilde{\Theta}}_f \Gamma_f^{-1} \widetilde{\Theta}_f^T\right\} \\ &+ \mathrm{Tr}\left\{e\phi(u)^T \widetilde{\Theta}_g^T + \dot{\widetilde{\Theta}}_g \Gamma_g^{-1} \widetilde{\Theta}_g^T\right\} - e^T\delta \\ \overset{(b)}{=}& -\lambda e^T e + \mathrm{Tr}\left\{\left(e\phi(y)^T \Gamma_f + \dot{\widehat{\Theta}}_f\right)\Gamma_f^{-1} \widetilde{\Theta}_f^T\right\} \\ &+ \mathrm{Tr}\left\{\left(e\phi(u)^T \Gamma_g + \dot{\widehat{\Theta}}_g\right)\Gamma_g^{-1} \widetilde{\Theta}_g^T\right\} - e^T\delta,\end{aligned}$$

where

- (a) follows by applying the property of trace $\phi(\cdot)^T \widetilde{\Theta}^T e = \mathrm{Tr}\{\phi(\cdot)^T \widetilde{\Theta}^T e\} = \mathrm{Tr}\{e\phi(\cdot)^T \widetilde{\Theta}^T\}$, and
- (b) follows by the fact that Θ_f^* and Θ_g^* are constants (i.e., $\dot{\widetilde{\Theta}}_f(t) \triangleq \dot{\widehat{\Theta}}_f(t)$ and $\dot{\widetilde{\Theta}}_g(t) \triangleq \dot{\widehat{\Theta}}_g(t)$).

First, consider the ideal case where the approximation error $\delta(t) = 0$. To obtain stable adaptive laws then $\dot{V}$ must be negative semi-definite. This can be done by simply selecting

$$\dot{\widehat{\Theta}}_f = -e\phi(y)^T\Gamma_f, \quad \text{and} \quad \dot{\widehat{\Theta}}_g = -e\phi(u)^T\Gamma_g, \tag{11}$$

which yield $\dot{V}(t) = -\lambda e e^T \leq 0$ and hence stability of the overall identification scheme is guaranteed. However, in the case where the approximation error $\delta(t) \neq 0$ then the adaptive laws given by (11) cannot prevent the undesirable scenario of $\dot{V} > 0$, and hence stability is not ensured. In that case, one must deal with the phenomenon known as *parameter drift* [15]. Several methods exist in the literature for preventing parameter drift, i.e., the projection algorithm, σ-modification, dead-zone, etc. [16]–[18]. When applied, the above modifications guarantee that $||\widehat{\Theta}_f(t)||_F \leq M_f$, $||\widehat{\Theta}_g(t)||_F \leq M_g$ for all $t \geq 0$ as long as $||\widehat{\Theta}_f(0)||_F \leq M_f$ and $||\widehat{\Theta}_g(0)||_F \leq M_g$.

When using the above adaptive laws, it is well known that $\widehat{\Theta}_f(t) \longrightarrow \Theta_f^*$, $\widehat{\Theta}_g(t) \longrightarrow \Theta_g^*$ if and only if $\phi(y(t))$ and $\phi(u(t))$ are persistently exciting signals.

Definition 2.2: A bounded vector signal $\phi(t) \in \mathbb{R}^n$ is persistently exciting (PE) in $\mathbb{R}^n$ if there exists $t_f > 0$ and $\gamma > 0$ such that

$$\int_t^{t+t_f} \phi(\tau)\phi(\tau)^T d\tau \geq \gamma I, \quad \text{for all } t \geq 0.$$

Next, two examples are presented to illustrate that in the absence of a PE signal, convergence of the output estimation error to zero does not necessarily imply that the parameter estimates will converge to their true values. In the first example, we consider a linear system with two unknown parameters, while in the second example (given in [14]), we consider a system with an unknown nonlinear function which is approximated using the online learning scheme and Lyapunov synthesis method.

Example 2.3: Consider the linear system

$$\dot{y} = -\zeta_1 y + \zeta_2 u, \tag{12}$$

where $u(t)$ is the input, $y(t)$ is the output, and ζ_1, ζ_2 are unknown parameters. For simulation purposes, take the unknown parameters to be equal to $\zeta_1 = 2$ and $\zeta_2 = 1$. Based on the online learning scheme and the Lyapunov synthesis method, we have that

$$\begin{aligned}\dot{y} &= \lambda y - \lambda y - \zeta_1 y + \zeta_2 u, \\ \dot{\widehat{y}} &= \lambda y - \lambda \widehat{y} - \widehat{\zeta}_1 y + \widehat{\zeta}_2 u,\end{aligned}$$

and the output estimation error is given by

$$\dot{e} = -\lambda e + \widetilde{\zeta}_1 y - \widetilde{\zeta}_2 u,$$

where $e = \widehat{y} - y$, $\widetilde{\zeta}_1 = \zeta_1 - \widehat{\zeta}_1$ and $\widetilde{\zeta}_2 = \zeta_2 - \widehat{\zeta}_2$. To apply the Lyapunov synthesis method, we choose the Lyapunov function to be equal to

$$V(e, \widetilde{\zeta}_1, \widetilde{\zeta}_2) = \frac{1}{2}e^2 + \frac{1}{2\gamma}\widetilde{\zeta}_1^2 + \frac{1}{2\gamma}\widetilde{\zeta}_2^2, \tag{13}$$

where γ and λ are design constants[2]. Taking the time derivative of (13), and after some manipulations, we have that

$$\dot{V} = -\lambda e^2 + \frac{\widetilde{\zeta}_1}{\gamma}\left(\gamma y e - \dot{\widehat{\zeta}}_1\right) - \frac{\widetilde{\zeta}_2}{\gamma}\left(\gamma u e + \dot{\widehat{\zeta}}_2\right).$$

For the function to qualify as a Lyapunov function, $\dot{V}$ must be negative and hence we obtain the following adaptive laws

$$\dot{\widehat{\zeta}}_1 = \gamma e y, \quad \text{and} \quad \dot{\widehat{\zeta}}_2 = -\gamma e u. \tag{14}$$

Fig. 1 depicts simulation results by considering two type of inputs, i.e., (i) $u(t) = \sin(5t)$ and (ii) $u(t)$ being the unit step function, and by choosing the design constants equal to $\lambda = 1$ and $\gamma = 10$. Note that, in the case of $u(t)$ being the unit step function (not a PE signal), even though the output estimation error (Fig. 1c) goes to zero, this does not necessarily imply that the parameter estimates $\widehat{\zeta}_1, \widehat{\zeta}_2$ converge to their true values (Fig. 1d).

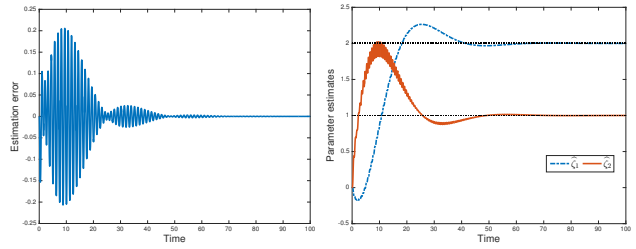
Example 2.4: [14] Consider the nonlinear dynamical system

$$\dot{y} = f(y) + u,$$

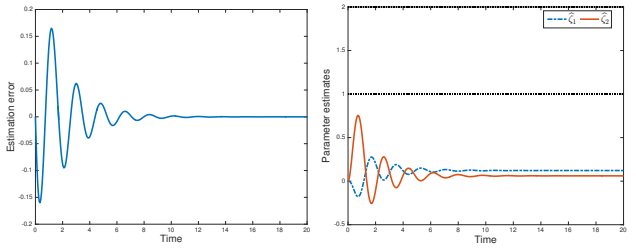
where $u(t)$ is the input, $y(t)$ is the output, and f is a nonlinear function that is assumed unknown. For simulation purposes, we take the unknown function f to be

$$f(y) = \exp(-y) - 1.$$

[2]In general γ is known as the adaptive gain (step size in optimization problems). If the adaptive gain is small, then adaptation and learning are slow. On the other hand, if the adaptive gain is large, then adaptation is faster. The design constant λ is related to the cut-off frequency ω_c of a low pass filter, $1/(s+\lambda)$. As the value of λ increases the cut-off frequency increases, which implies that the filter attenuates (reduces the amplitude of) the input signal for frequencies higher than the cut-off.

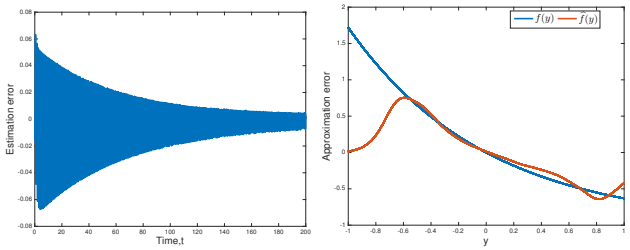


(a) $u(t) = \sin(5t)$ (b) $u(t) = \sin(5t)$

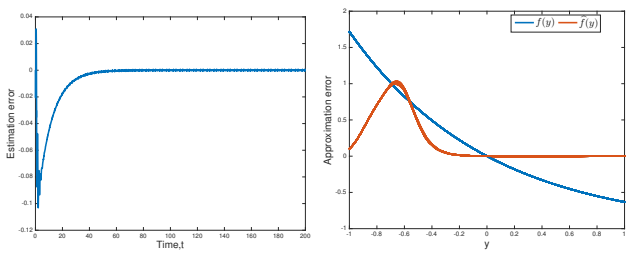


(c) $u(t) =$ unit step (d) $u(t) =$ unit step

Fig. 1: Simulation results for Example 2.3. The top two plots show the results for $u(t) = \sin(5t)$, while the bottom two show the results for $u(t)$ being a unit step function. The left plots show the output estimation error $e = \widehat{y} - y$ and the right plots show the parameter estimates $\widehat{\zeta}_1, \widehat{\zeta}_2$.



(a) $u(t) = 5\sin(2\pi t)$ (b) $u(t) = 5\sin(2\pi t)$



(c) $u(t) = 5\sin(2\pi t)\exp(-t)-1$ (d) $u(t) = 5\sin(2\pi t)\exp(-t)-1$

Fig. 2: Simulation results for Example 2.4. The top two plots show the results for $u(t) = 5\sin(2\pi t)$, while the bottom two show the results for $u(t) = 5\sin(2\pi t)\exp(-t)-1$. The left plots show the output estimation error and the right plots show the approximation error.

By employing the online learning scheme and Lyapunov synthesis method, we have that

$$\dot{y} = \lambda y - \lambda y + \widehat{f}(y;\theta^*) + u - \delta,$$
$$\dot{\widehat{y}} = \lambda y - \lambda \widehat{y} + \widehat{f}(y;\widehat{\theta}) + u,$$

where $\widehat{f}(y;\theta^*) = \sum_{i=1}^{N} \theta_i^* \phi_i(y)$ and $\widehat{f}(y;\widehat{\theta}) = \sum_{i=1}^{N} \widehat{\theta}_i \phi_i(y)$. The output estimation error is given by

$$\dot{e} = -\lambda e + \sum_{i=1}^{N} \widetilde{\theta}_i \phi_i(y) + \delta,$$

where $e = \widehat{y} - y$ and $\widetilde{\theta} = \widehat{\theta} - \theta^*$. Choosing a Lyapunov function to be equal to

$$V(e,\widetilde{\theta}) = \frac{1}{2}e^2 + \sum_{i=1}^{N} \frac{1}{2\gamma_i}\widetilde{\theta}_i^2, \tag{15}$$

and taking its time derivative, we have that

$$\dot{V} = -\lambda e^2 + \sum_{i=1}^{N} \frac{1}{\gamma_i}\widetilde{\theta}_i\Big(\gamma_i e \phi_i(y) + \dot{\widehat{\theta}}_i\Big) + \delta e.$$

The resulting adaptive laws for $\widehat{\theta}_i$ are given by

$$\dot{\widehat{\theta}}_i = \gamma_i e \phi(y), \quad i = 1, \ldots, N. \tag{16}$$

We employ an RBF network with $N = 12$ basis functions of the form $\phi_i(y) = \exp(-|y - c_i|^2/\sigma^2)$, where the centers are fixed and uniformly distributed between $[-1, 1]$ and the width is equal to $\sigma = \sqrt{2}/10$. Fig. 2 depicts the simulation results by considering two types of inputs, i.e., (i) $u(t) = 5\sin(2\pi t)$ and (ii) $u(t) = 5e^{-t}\sin(2\pi t) - 1$ (not a PE signal), and by choosing the design constants equal to $\lambda = 10$ and $\gamma_i = 1$ for all $1 \leq i \leq 12$. As seen from the plots, in both input cases the estimation error (left plots) converges towards zero, however the approximation error (right plots) is very small only for the first input case.

The above examples clearly illustrate the issue of persistency of excitation. We note that even though the estimation error goes to zero, this does not necessarily imply that the parameter estimates converge to their true values; moreover, the approximation error becomes small only in the region in which the input to the approximator varies.

III. Biologically Inspired Application

In this section we apply the theoretical results presented in the previous section to a biologically inspired application, in which data are obtained from input-output measurements. First, we introduce a typical gene regulatory network model.

A. Dynamical model of gene regulatory networks

A gene regulatory network indicates how different genes impact each other and can be represented as a directed graph $G = (V, E)$, with vertices or nodes $V = \{v_1, \ldots, v_n\}$ corresponding to genes and directed edges E corresponding to regulatory interactions. A directed edge from gene i to gene j indicates that the product of gene i influences the expression of gene j either by "promoting" ($x_i \to x_j$) or by "inhibiting" it ($x_i \dashv x_j$). In particular, the expression of any single gene i can be written as a linear combination of functions of different genes which act as the regulators or inputs to the model. These regulatory functions are

monotonically increasing or decreasing, and usually take the Michaelis-Menten or Hill-type forms.

The underlying dynamical system of the gene regulatory network is described by

$$\begin{aligned}\dot{y}_i(t) &= -\alpha_i y_i(t) + g_i(y_{j_1}(t), \ldots, y_{j_n}(t)) \\ &= -\alpha_i y_i(t) + \sum_j |\rho_{ij}| \bar{g}_{ij}(y_j(t)),\end{aligned} \tag{17}$$

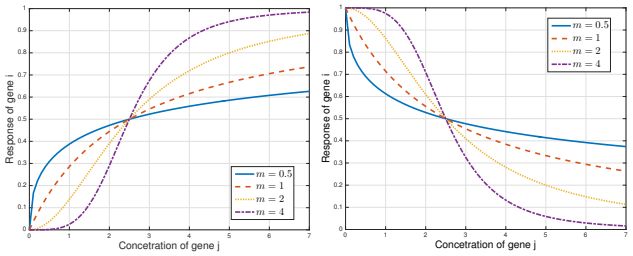
where it is assumed that the (total) nonlinearity $g_i(\cdot)$ can be decomposed and expressed as the sum of individual nonlinearities $\bar{g}_{ij}(\cdot)$. Here $y_i(t)$ denotes the concentration of gene i at time t, $\dot{y}_i(t)$ denotes the change in concentration of the i-th gene, variable α_i denotes the degradation rate of gene i, parameters ρ_{ij} denote the regulation strength of gene j on gene i. In particular, $\rho_{ij} > 0$ corresponds to a positive regulation (activation), $\rho_{ij} < 0$ corresponds to a negative regulation (inhibition), and $\rho_{ij} = 0$ corresponds to no regulation of gene i by gene j. The regulatory functions $\bar{g}_{ij}(\cdot)$ are monotonic functions of Hill-type form, that is, if gene j is an activator for gene i, then

$$\bar{g}_{ij}(y_j(t)) = \left(\frac{\left(\frac{y_j(t)}{\beta_{ij}}\right)^{m_{ij}}}{1 + \left(\frac{y_j(t)}{\beta_{ij}}\right)^{m_{ij}}} \right), \quad \text{for } \rho_{ij} > 0 \tag{18}$$

and if gene j is an inhibitor for gene i, then

$$\bar{g}_{ij}(y_j(t)) = \left(1 - \frac{\left(\frac{y_j(t)}{\beta_{ij}}\right)^{m_{ij}}}{1 + \left(\frac{y_j(t)}{\beta_{ij}}\right)^{m_{ij}}} \right), \quad \text{for } \rho_{ij} < 0. \tag{19}$$

Here, m_{ij} are the Hill coefficients, and β_{ij} are the threshold



(a) Activation of gene i by gene j (b) Inhibition of gene i by gene j

Fig. 3: Hill function for different values of Hill coefficients $m = 0.5, 1, 2, 4$ and threshold parameter $\beta = 2.5$.

parameters. Fig. 3 shows Hill functions for different values of Hill coefficients. As an example, consider the network model shown in Fig. 4. This network topology corresponds to a set of ordinary differential equations (ODE's) given by (27). Here gene 1 is activated by gene 5, gene 2 is activated by gene 1, gene 3 is activated by gene 1 and gene 2, gene 4 is activated by gene 1 and inhibited by gene 3, and gene 5 is activated by gene 4 and inhibited by gene 2. It is clear from this example that to apply the learning scheme and Lyapunov synthesis method of Section II-B, one first needs to transform the set of ODE's into an equivalent input-output form.

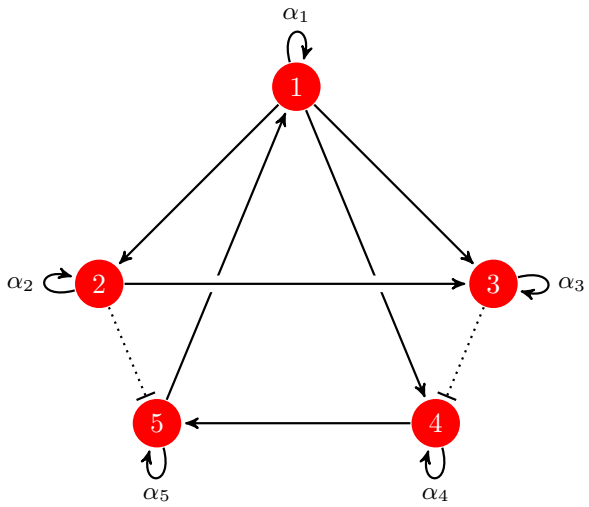


Fig. 4: Network model.

In general, the gene regulatory network can be transformed into an input-output system by considering the effect of different genes on gene i as the inputs or regulators to the system. In particular, (17) can be re-written as

$$\begin{aligned}\dot{y}_i(t) &= -\alpha_i y_i(t) + g_i(u_{j_1}(t), \ldots, u_{j_n}(t)) \\ &= -\alpha_i y_i(t) + \sum_j |\rho_{ij}| \bar{g}_{ij}(u_j).\end{aligned} \tag{20}$$

Note that, the input u and the output y are available for measurement, but $\dot{y}$ is not. One way to obtain $\dot{y}$ is to interpolate y and use numerical derivatives however this is not desirable. To avoid the use of differentiators, the proposed methodology applies instead some filtering techniques. Note that, in what follows we use the notation $\frac{1}{s+\lambda}(x)$ to express the output of the filter using $x(t)$ as the input. Specifically, by filtering (20) with a stable first-order filter $1/(s+\lambda)$, where $\lambda > 0$, we obtain[3]

$$y_i(t) = \frac{1}{s+\lambda}\Big(-\alpha_i y_i(t) + \widehat{g}_i(z(t); \theta_g^*) + \lambda y_i(t)\Big) + \delta_f(t), \tag{21}$$

where $z(t) = (u_{j_1}(t), \ldots, u_{j_n}(t))$ and $\delta_f(t)$ is the filtered approximation error, i.e., $\delta_f = \frac{1}{s+\lambda}(\delta)$. Based on (21), the learning model is given by

$$\widehat{y}_i(t) = \frac{1}{s+\lambda}\Big(-\widehat{\alpha}_i(t) y_i(t) + \widehat{g}_i(z(t); \widehat{\theta}_g(t)) + \lambda y_i(t)\Big). \tag{22}$$

Notice that the gene regulatory network in (20) has now been transformed into the form of the parametric model (7) and the learning model (8), and hence the learning scheme and Lyapunov synthesis method of Section II-B can be directly applied.

Remark 3.1: In the gene regulatory network (20), α_i are unknown parameters and $g_i(\cdot)$ are unknown nonlinear functions that need to be estimated. In many cases, the designer also needs to include additional biological knowledge into the model, to study its response and to design additional

[3]Depending on the application, filters of higher order can also be applied.

experiments. This partial *a priori* information can also be included into the model. In particular, to formulate such a scenario assume that the nonlinear function g is partially known and can be decomposed into two parts, i.e., $g_i(z(t)) = g_0(z(t)) + \widetilde{g}_i(z(t))$, namely, the known (or nominal) part $g_0(\cdot)$ and the unknown part $\widetilde{g}_i(\cdot)$. Then (20), in its general form, can be written as follows.

$$\dot{y}_i(t) = -\alpha_i y_i(t) + g_0(z(t)) + \widetilde{g}_i(z(t)). \tag{23}$$

By filtering (23) a parametric model is given by

$$\xi_i(t) = \frac{1}{s+\lambda}\Big(-\alpha_i y_i(t) + \widehat{g}_i(z(t);\theta_g^*)\Big) + \delta_f(t), \tag{24}$$

where $\xi(t)$ is a measurable variable computed as[4]

$$\xi_i(t) = \frac{s}{s+\lambda}\Big(y_i(t)\Big) - \frac{\lambda}{s+\lambda}\Big(g_0(z(t))\Big), \tag{25}$$

and $\delta_f(t)$ is given by

$$\delta_f(t) = \frac{1}{s+\lambda}\Big(\widetilde{g}_i(z(t)) - \widehat{g}_i(z(t);\theta_g^*)\Big). \tag{26}$$

The learning scheme of Section II-B can be derived in a similar fashion. The architecture of the error learning scheme is depicted in Fig. 5.

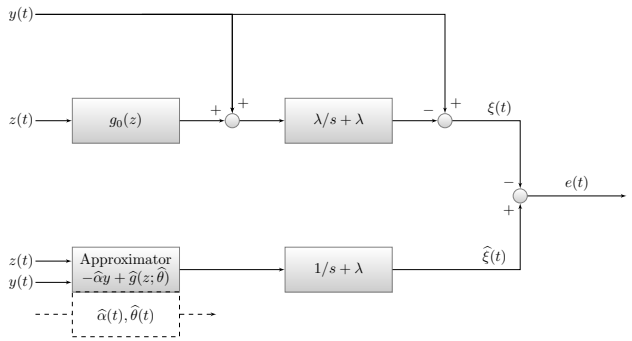


Fig. 5: Block diagram of the error learning scheme. The dashed box under the approximator indicates the dynamics of the parameter estimator.

B. Illustrative example

To get greater insight on how the error filtering learning scheme can be applied to estimate the unknown nonlinearities and to identify the true network that generated the data, we illustrate the reconstruction of the five-gene regulatory network of Fig. 4.

Data was simulated using the differential equation model[5] (27), with degradation rates $\alpha_1 = \alpha_2 = \alpha_3 = 0.9$ and $\alpha_4 = \alpha_5 = 1.5$ for the five genes. The threshold parameter of the Hill functions was set to $\beta = 1.5$, with Hill coefficient $m = 8$. The parameters for positive and negative regulation strength were set equal to $\rho_{15} = \rho_{21} = \rho_{31} = \rho_{32} = \rho_{41} = \rho_{54} = 2$, $\rho_{43} = \rho_{52} = -2$, and zero if there is no interaction between gene i and gene j. By numerical integration of the ODE model (27) in Matlab, using the function *ode45*, the concentration of the five genes is depicted in Fig. 6.

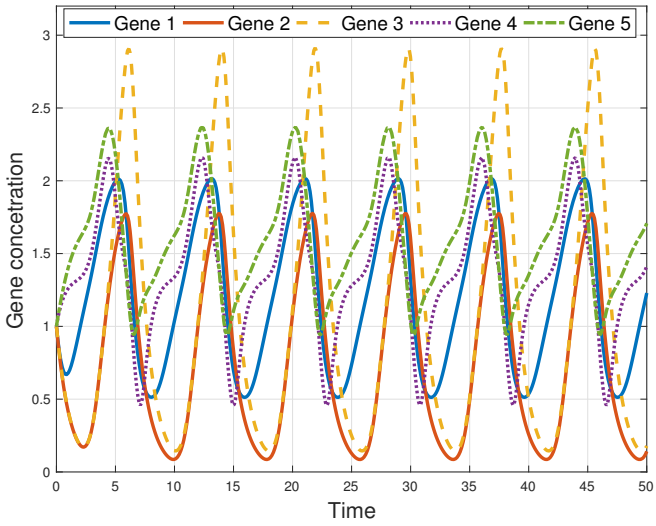


Fig. 6: Simulated data of the five-gene network model.

$$\dot{y}_1 = -\alpha_1 y_1 + |\rho_{15}|\frac{u_5^m}{\beta^m + u_5^m} \tag{27a}$$

$$\dot{y}_2 = -\alpha_2 y_2 + |\rho_{21}|\frac{u_1^m}{\beta^m + u_1^m} \tag{27b}$$

$$\dot{y}_3 = -\alpha_3 y_3 + |\rho_{31}|\frac{u_1^m}{\beta^m + u_1^m} + |\rho_{32}|\frac{u_2^m}{\beta^m + u_2^m} \tag{27c}$$

$$\dot{y}_4 = -\alpha_4 y_4 + |\rho_{41}|\frac{u_1^m}{\beta^m + u_1^m} + |\rho_{43}|\frac{\beta^m}{\beta^m + u_3^m} \tag{27d}$$

$$\dot{y}_5 = -\alpha_5 y_5 + |\rho_{52}|\frac{\beta^m}{\beta^m + u_2^m} + |\rho_{54}|\frac{u_4^m}{\beta^m + u_4^m}. \tag{27e}$$

1) Procedure followed: The learning scheme and Lyapunov synthesis method is applied to reconstruct the five-gene regulatory network by assuming that all genes are accessible for measurement and also that enough input-output (steady-state) data are available for simulation purposes. In particular, data was generated by numerically integrating the differential equations (27) for a horizon $t_0 = 0$ to $t_f = 500$; however, only a portion of the data is used for the approximation method while the remaining data are used for model validation. Specifically, 70% data are employed for training and 30% for testing.

It is worth mentioning that in many cases, even though the estimation error goes to zero (i.e., see Fig. 1(c) or/and Fig. 2(c)), this does not necessarily imply that the approximation error goes to zero or that the parameter estimates converge to their true values (i.e., see Fig. 1(d) or/and Fig. 2(d)). This issue is related to the concept of persistent excitation, as we already discussed in Section II-B. Our primary objective here is to choose the right pairs of genes, and consequently reconstruct the true gene regulatory network that generated the data. Towards this end, we employ the function *odextend* in Matlab using the remaining 30% data for testing. If the estimated parameter and the approximation function are

[4]It is noted that in deriving (25), we use the fact that $\dot{y}(t) = s(y(t))$ assuming zero initial conditions.

[5]For ease of notation the time dependence of $y(t)$ and $u(t)$ is dropped.

the true ones, then they should be able to reproduce the remaining data, i.e., $y(t)$ for $t = 7t_f/10$ till $t = t_f$. Hence, the model validation can be evaluated in terms of a fitness criterion. In particular, as a fitness criterion for any gene i, we use the normalized mean square error given by

$$\text{fitness}(i) = 1 - \frac{\sum_{t=7t_f/10}^{t_f} \Big(y_i(t) - \widehat{y}_i(t)\Big)^2}{\sum_{t=7t_f/10}^{t_f} \Big(y_i(t) - mean(y_i)\Big)^2}, \quad (28)$$

taking values from $-\infty$ (bad fit) to 1 (perfect fit). The procedure followed consists of the following steps.

Step 1: By following an all-to-all approach, we test all genes pairwise as shown in Fig. 7, by using N basis functions of the Gaussian form (5),

$$\phi_j(u) = \exp\Big(-|u - c_j|^2/\sigma_j^2\Big), \ j = 1, 2, \ldots, N.$$

Since, in general, it is difficult to know in advance the number of the (network) centers which are required to reflect the allocation of the input data, we run our simulations for a range of values of $N = 2, \ldots, m$. For each N, the centers are chosen using the k-means clustering algorithm with the widths $\sigma_j = \sigma$ set equal to the mean value of the pairwise distance between the (k-means) resulting centers. It should be pointed out that the k-means clustering algorithm often reaches a solution which depends on the starting cluster center positions, i.e., choosing a different set of starting positions might lead to a different/local solution. To overcome this problem and select a unique/global solution, the k-means clustering algorithm was applied on the input data several times using several initializations of cluster center positions. In addition, to keep the comparison fair, the design constants were set equal to $\lambda = 1$ and $\gamma_i = 1$ for all $1 \leq i \leq N$.

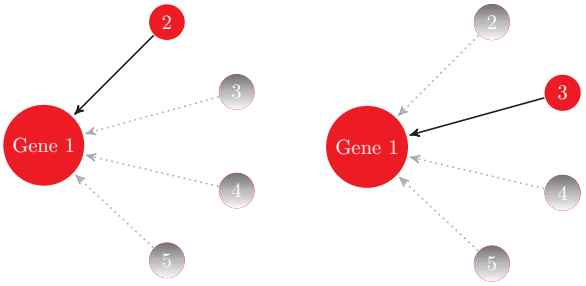


Fig. 7: All-to-all approach of Step 1. In the left plot, gene 1 is acting as the output and gene 2 as the input. In the right plot, gene 1 is acting as the output and gene 3 as the input.

As an example, in Table I we report the values using gene 1 as the output and gene 5 as the input for each of N basis functions. In particular, Table I displays the number of basis functions $N = 1, 2, \ldots, 7$, the corresponding value of the width σ, and the fitness results given by (28). The best fitness result is highlighted. Similar fitness results were also obtained during the comparison between all pairs of genes, for N large enough, but are not shown here due to space limitations. Instead, a detailed comparison including the best fitness results (only), for each pair of genes, is presented in Table II. In particular, Table II displays the best fitness results and the corresponding number of basis functions (in parenthesis) for each possible pair of genes $(k, l), k \neq l, k, l = 1, 2, \ldots, 5$, with k denoting genes acting as the output (rows) and l denoting genes acting as the input (columns). The best result within a row, which is also above a validation threshold of 0.9, is highlighted. It is noticeable that the pairs of genes $(1, 5)$ and $(2, 1)$, which correspond to a true link in the network model, provide the best fitness results which are also above the validation threshold. For all remaining genes, i.e., gene 3, 4 and 5 (acting as outputs), for which the fitness result is below 0.9 we proceed to the next step.

N	σ	fitness(1)
2	0.985	0.966
3	0.943	0.905
4	0.874	0.852
5	0.849	0.775
6	0.799	0.716
7	0.788	0.689

TABLE I: Fitness results using gene 1 as the output and gene 5 as the input.

Output \ Input	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
Gene 1		-0.042 (2)	-1.07 (2)	0.845 (7)	0.966 (2)
Gene 2	0.949 (7)		-0.620 (2)	0.137 (3)	0.514 (5)
Gene 3	0.723 (4)	0.607 (4)		0.146 (3)	0.057 (2)
Gene 4	-0.091 (2)	-5.170 (2)	-1.670 (2)		-0.190 (3)
Gene 5	-8.541 (2)	0.310 (8)	0.268 (8)	-2.520 (4)	

TABLE II: All-to-all approach fitness results.

Step 2: As in the previous step, we follow an all-to-all approach for the remaining genes; however, at this step we use as inputs pairs of genes. We employ N basis functions of the Gaussian form,

$$\phi_j(u_1, u_2) = \exp\Big(-(|u_1 - c_{1,j}|^2/\sigma_{1,j}^2 + |u_2 - c_{2,j}|^2/\sigma_{2,j}^2)\Big),$$

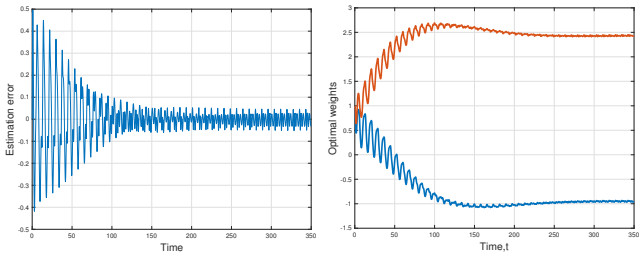
where $j = 1, \ldots, N$. Again, for each N, the centers are chosen using the k-means clustering algorithm with the widths $\sigma_{1,j} = \sigma_1$, $\sigma_{2,j} = \sigma_2$ set equal to the mean value of the pairwise distance between the (k-means) resulting centers and the design constants were set equal to $\lambda = 10$ and $\gamma_i = 10$ for all $1 \leq i \leq N$.

By following the same approach as in Step 1, in Table III we display the best fitness results and the corresponding number of basis functions for all remaining genes, i.e., genes 3, 4 and 5. The best result within a row which is also above the validation threshold is highlighted. It is interesting to note that genes which correspond to a true link in the network model give the best fitness results with the lower number of basis functions.

Simulation results obtained using gene 1 as the output and gene 5 as the input, a true link in the network model, are

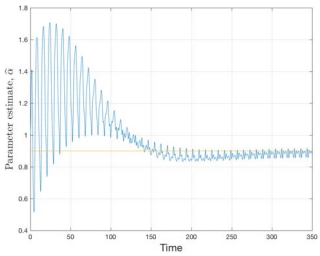
Output	Input					
	(1,2)	(1,4)	(1,5)	(2,4)	(2,5)	(4,5)
GENE 3	0.973 (7)	0.926 (12)	0.968 (11)	0.967 (12)	0.963 (10)	0.302 (2)
	(1,2)	(1,3)	(1,5)	(2,3)	(2,5)	(3,5)
GENE 4	0.881 (12)	0.949 (9)	0.637 (12)	-1.217 (11)	0.818 (6)	0.784 (12)
	(1,2)	(1,3)	(1,4)	(2,3)	(2,4)	(3,4)
GENE 5	0.915 (12)	0.862 (5)	0.805 (12)	0.498 (3)	0.931 (7)	0.553 (12)

TABLE III: All-to-all approach fitness results using as inputs pairs of genes.

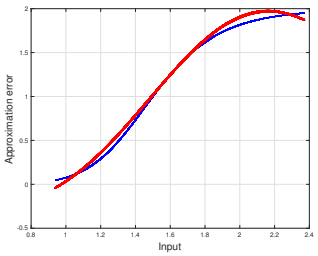


(a) Estimation error

(b) Optimal weights



(c) Parameter estimate



(d) Approximation error

Fig. 8: Five-gene network model. Simulation results obtained by the pair of genes 1 (output) and 5 (input).

depicted in Fig. 8. The top two plots, Fig. 8(a)-(b), depict the estimation error and the optimal weights, respectively. The bottom two plots, Fig. 8(c)-(d), depict the estimate of the unknown parameter α_1 converging to its true value, and the approximation error, respectively. The approximation plot (Fig. 8(d)) depicts the function $\bar{g}_{15}(u_5)$ (blue thin line) and its approximator $\widehat{g}(u_5;\widehat{\theta}_g)$ (red thick line), which denotes the approximation function at the end of the simulation, i.e., $t = 7t_f/10 = 350$, at which point the optimal weights $\widehat{\theta}_g$ have converged to their final values.

IV. CONCLUSION

The principal contribution of this paper is the introduction of an error filtering learning scheme and a Lyapunov synthesis method for reconstructing gene regulatory networks from input-output data based on nonlinear differential equations. The unknown nonlinearities are modelled by employing radial basis functions networks and the model validation is performed by taking advantage of the so-called persistency of excitation condition. Results illustrate that the proposed methodology can be used to adequately reconstruct gene regulatory networks based on input-output data. Future directions include the extension of the results presented in this paper by developing effective methods to deal with noisy and partial measurements.

REFERENCES

[1] N. Dojer, A. Gambin, A. Mizera, B. Wilczynski, and J. Tiuryn, "Applying dynamic Bayesian networks to perturbed gene expression data." *BMC Bioinformatics*, vol. 7, p. 249, 2006.

[2] S. Liang, S. Fuhrman, and R. Somogyi, "REVEAL, a general reverse engineering algorithm for inference of genetic network architectures," in *Pacific Symposium on Biocomputing*, vol. 3, 1998, pp. 18–29.

[3] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. D. Favera, and A. Califano, "Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context." *BMC Bioinformatics*, vol. 7, no. S-1, 2006.

[4] T. Chen, H. L. He, and G. M. Church, "Modeling gene expression with differential equations." in *Pacific Symposium on Biocomputing*, R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, Eds., 1999, pp. 29–40.

[5] T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins, "Inferring genetic networks and identifying compound mode of action via expression profiling," *Science*, vol. 301, no. 5629, pp. 102–105, 2003.

[6] Y. Yuan, G.-B. Stan, S. Warnick, and J. Goncalves, "Robust dynamical network structure reconstruction," *Automatica*, vol. 47, no. 6, pp. 1230 – 1235, 2011, special Issue on Systems Biology.

[7] D. P. Hayden, Y. Yuan, and J. M. Goncalves, "Robust network reconstruction in polynomial time." in *Proceedings of IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 4616–4621.

[8] J. M. 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.

[9] K. J. Astrom and B. Wittenmark, *Adaptive Control*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1994.

[10] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[11] E. Sontag, A. Kiyatkin, and B. N. Kholodenko, "Inferring dynamic architecture of cellular networks using time series of gene expression, protein and metabolite data," *Bioinformatics*, vol. 20, no. 12, p. 1877, 2004.

[12] D. Marbach, J. C. Costello, R. Küffner, N. M. Vega, R. J. Prill, D. M. Camacho, K. R. Allison, M. Kellis, J. J. Collins, G. Stolovitzky *et al.*, "Wisdom of crowds for robust gene network inference," *Nature methods*, vol. 9, no. 8, pp. 796–804, 2012.

[13] G. Golub and C. Van Loan, *Matrix Computations*, ser. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996.

[14] J. A. Farrell and M. M. Polycarpou, *Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches (Adaptive and Learning Systems for Signal Processing, Communications and Control Series)*. Wiley-Interscience, 2006.

[15] P. A. Ioannou and P. V. Kokotovic, *Adaptive Systems with Reduced Models*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1983.

[16] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence, and Robustness*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[17] G. C. Goodwin and D. Q. Mayne, "A parameter estimation perspective of continuous time model reference adaptive control," *Automatica*, vol. 23, no. 1, pp. 57–70, Jan. 1987.

[18] P. Ioannou and K. Tsakalis, "A robust direct adaptive controller," *IEEE Transactions on Automatic Control*, vol. 31, no. 11, pp. 1033–1043, Nov. 1986.