Interpretable Design of Reservoir Computing Networks Using Realization Theory

Wei Miao, *Member, IEEE*, Vignesh Narayanan, *Member, IEEE*, and Jr-Shin Li, *Senior Member, IEEE*

***Abstract*— The reservoir computing networks (RCNs) have been successfully employed as a tool in learning and complex decision-making tasks. Despite their efficiency and low training cost, practical applications of RCNs rely heavily on empirical design. In this article, we develop an algorithm to design RCNs using the realization theory of linear dynamical systems. In particular, we introduce the notion of α-stable realization and provide an efficient approach to prune the size of a linear RCN without deteriorating the training accuracy. Furthermore, we derive a necessary and sufficient condition on the irreducibility of the number of hidden nodes in linear RCNs based on the concepts of controllability and observability from systems theory. Leveraging the linear RCN design, we provide a tractable procedure to realize RCNs with nonlinear activation functions. We present numerical experiments on forecasting time-delay systems and chaotic systems to validate the proposed RCN design methods and demonstrate their efficacy.**

***Index Terms*— Control systems, realization theory, recurrent neural networks (RNNs), reservoir computing networks (RCNs), time-series forecasting.**

I. Introduction

THE reservoir computing network (RCN) is a biomimetic computational tool that is increasingly used in a variety of applications to solve complex decision-making problems [1]–[3]. Essentially, the RCN is a class of recurrent neural networks (RNNs), which is composed of one hidden layer, typically with a large number of sparsely interconnected neurons, and a linear output layer. In contrast to the classical RNN, a distinct feature of the RCN is that all of its connections in the hidden layer are randomly predetermined and fixed. Hence, the training process of the RCN involves only learning the weights of its linear output layer in a supervised learning framework.

The existing supervised learning approach to training an RCN was proposed in [1]. Subsequently, the RCN was successfully employed for forecasting time series with applications in finance [4], [5], wireless communication [6], speech recognition [7], and robot navigation [8]. Notwithstanding its efficient training procedure, the major limitation of the RCN lies in the fact that its practical application relies heavily on the empirical design of the hyperparameters of the network, including its size [9].

Recently, there has been a renewed interest in developing tractable methods for designing neural networks that are suitably deployed in diverse scenarios [10], [11]. In this context, deriving rigorous and systematic techniques to design neural networks, especially establishing principled strategies for selecting their hyperparameters that yield the desired performance, is compelling but challenging. In this article, we propose a tractable approach to design RCNs that warrant effective functioning for given datasets. In particular, we focus on the application of RCNs to learn dynamic models of dynamical systems from their time-series measurement data and develop rigorous design principles to prune the number of hidden-layer nodes in RCNs without deteriorating the training performance. Leveraging the notions of controllability and observability, we derive a necessary and sufficient condition on irreducibility of RCNs with linear activation functions. This in turn results in an interpretable RCN pruning algorithm, where the controllability and observability matrices of an RCN inform on its size and irreducibility Furthermore, we illustrate that the developed irreducible linear realization of the RCN not only sufficiently represents the underlying dynamics inherited in the time-series data but also contributes to a tractable design of general RCNs with nonlinear activation functions.

This article is organized as follows. In Section II, we provide a brief review of related works that motivate the need of our developments. In Section III, through tailoring existing results on the learnability of RNNs, we motivate our realization-theoretic RCN design principles by illustrating how an RCN achieves τ-step ahead forecast of time series associated with a dynamical system. In Section IV, we introduce realization-theoretic aspects from systems theory to facilitate a comprehensive RCN design and then establish a necessary and sufficient condition on the irreducibility of a linear RCN that achieves the desired training accuracy. This result in turn forms the basis for an educated design of RCNs with nonlinear activation functions. In Section V, we present several results using an RCN to forecast time series and learn chaotic systems to demonstrate the applicability of the proposed RCN design framework.

Manuscript received September 1, 2020; revised April 8, 2021 and September 23, 2021; accepted December 9, 2021. This work was supported in part by the National Science Foundation under Award CMMI-1933976 and Award CMMI-1763070, and in part by the NIH Grant R01GM131403-01. *(Corresponding author: Jr-Shin Li.)*

Wei Miao and Jr-Shin Li are with the Department of Electrical and Systems Engineering, Washington University in St. Louis, St. Louis, MO 63130 USA (e-mail: weimiao@wustl.edu; jsli@wustl.edu).

Vignesh Narayanan is with the AI Institute, University of South Carolina, Columbia, SC 29208 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3136495.

Digital Object Identifier 10.1109/TNNLS.2021.3136495

II. Related Works and Motivation

In this section, we briefly review the existing works on RCNs and point out the specific problems that we address in this article.

The computational framework of an RCN and its training procedure was first proposed in [1], where two conditions were hypothesized as requirements for successful applications of the RCNs—the echo state property (ESP) and a general compactness assumption on the training signal. In addition, a mathematical definition of the ESP was also provided in [12] and [13]. Intuitively, the ESP implies that the state of the RCN is uniquely determined by its input history rather than the initial condition of the network.

Thereafter, several results explaining the principles of the RCN, especially in evaluating some of its features, such as the ESP [14], [15], the memory capacity [16]–[18], and the stability [19], have been reported. In addition to investigating the fundamental properties of the RCN, multiple attempts addressing its design were also reported. The performance of the RCN in relation to the complexity of its network topology was analyzed in [20]. Using the interpretation of contracting maps, a discussion on the architecture of the RCN was presented in [21]. Supported by heuristic analyses, it was shown in [22] that the small-world network topology improved the performance (e.g., forecasting accuracy or memory capacity) of the RCN. Despite the prescribed results, existing applications using RCNs rely on randomly generated connection matrices. Using the random matrix theory, an explanation on why such a design of the RCN, in general, leads to acceptable performance was presented [23].

Recently, there has been a rising tide of interest in analyzing RCNs, especially RCNs with linear activation functions, using control-theoretic approaches [24], [25]. For instance, the connectivity patterns of RCNs were studied in [26] leveraging the concept of controllability matrix. Furthermore, it was proven in [27] that the memory capacity of a linear RCN can be characterized by the rank of its controllability matrix. Nevertheless, due to the inherited randomness and usage, the design process of the RCNs is still based on empirical strategies, and designing an RCN with minimum size to ensure the desired performance is compelling but remains elusive.

In this work, we focus on establishing schematic design principles to realize RCNs with linear and nonlinear activation functions, which achieves guaranteed training accuracy. The main contributions of this article include: 1) establishing the notion of α-stable realization for designing the weight/connection matrix in an RCN with desired training accuracy for a given dataset; 2) devising an algorithm based on realization theory to prune the size of linear RCNs with quantifiable training accuracy; and 3) deriving tractable guidelines for configuring RCNs with nonlinear activation functions through the linear RCNs obtained via irreducible realization.

Here, we adopt the definition of the ESP as introduced in [1], that is, an RCN is said to have the ESP if the state variables of the RCN are uniquely determined by the input history, regardless of the initial condition. Throughout this article, we denote $u[a; b]$ as a sequence with index starting from a to b, where $a < b$, i.e., $u[a; b] := \{u[a], \ldots, u[b]\}$.

III. Role of Takens Embedding in RCN Frameworks

In this section, we provide details of the RCN dynamics and its training procedure. We tailor existing results on the learnability of RNNs, in particular, the Takens embedding theorem, to render a comprehensive analysis of how an RCN learns the underlying dynamics of a time series. Based on the analyses, we discuss the applications of RCNs for the time-series forecasting problem, which motivates the design of RCNs using linear realization theory in Section IV. We begin with a brief introduction to the Takens theorem and discuss its role in understanding RCNs.

A. Takens Theorem and Its Implication on Time-Series Forecasting

We consider a time-dependent variable $s(t)$, evolving on an m-dimensional manifold $M \subset \mathbb{R}^p$, following the dynamics $\dot{s}(t) = F_s(s(t))$, where $F_s : M \to \mathbb{R}^p$ is a smooth vector field. Let $\upsilon : M \to \mathbb{R}$ be an observation function, and in practice, we measure a discrete sequence of observations, say $\upsilon[s(t_i)]$, where t_i for $i = 0, 1, 2, \ldots$ denoting the sampling instants. We can then define the propagation map $\phi : M \to M$ describing the flow of $s(t)$ at time t_i by $s(t_{i+1}) = \phi(s(t_i))$. Now, let $D(M) \subset C^2(M)$ denote the collection of functions such that, for any $f \in D(M)$, $f : M \to M$ has an inverse function $f^{-1} \in C^2(M)$, where $C^2(M)$ denotes the class of functions over M for which the first- and second-order derivatives are continuous. Then, for the dynamical system describing the time evolution of $s(t)$, we have $\phi \in D(M)$. The Takens theorem can be stated as follows.

Theorem 1 (Takens Theorem [28]): Let M be a compact manifold of dimension m. For pairs (ϕ, υ) with $\phi \in D(M)$ and $\upsilon \in C^2(M, \mathbb{R})$, it is a generic property that the map $\Phi_{\phi,\upsilon,2m+1} : M \to \mathbb{R}^{2m+1}$, defined by $\Phi_{\phi,\upsilon,2m+1}(s) = (\upsilon(s), \upsilon(\phi(s)), \ldots, \upsilon(\phi^{2m}(s)))$ is an embedding, where "generic" means open and dense in the C^1 topology.

From here on, we use Φ_{2m+1} as an abbreviation for $\Phi_{\phi,\upsilon,2m+1}$ and call $2m + 1$ as the "length" of the time-delay embedding Φ_{2m+1}. Intuitively, the Takens theorem states that for almost all pairs (ϕ, υ) defined on a compact manifold M of dimension m, there is an 1–1 correspondence from M to $\mathbb{R}^{2m+1}$ that preserves the structure of M. If the Takens theorem holds, then by the definition of an embedding, the inverse function for the map Φ_{2m+1} is well-defined. Hence, we can define a map $\psi_{2m+1} := \Phi_{2m+1} \circ \phi \circ \Phi^{-1}_{2m+1}$, which describes the same dynamical system as ϕ does, under a coordinate change of Φ_{2m+1}. The nontriviality of the construction of ψ_{2m+1} is that it forecasts a new observation when provided with the time-delayed observations $(\upsilon(s), \upsilon(\phi(s)), \ldots, \upsilon(\phi^{2m}(s)))$. Namely, it holds that $\psi_{2m+1}(\upsilon(s), \upsilon(\phi(s)), \ldots, \upsilon(\phi^{2m}(s))) = (\upsilon(\phi(s)), \upsilon(\phi^2(s)), \ldots, \upsilon(\phi^{2m+1}(s)))$, which implies that if one learns the explicit representation of ψ_{2m+1}, then the new observation, i.e., $\upsilon(\phi^{2m+1}(s))$, can be predicted based on the historical observations, $(\upsilon(s), \ldots, \upsilon(\phi^{2m}(s)))$. Specifically, if $\upsilon = I_d$, then ψ_{2m+1} essentially predicts how the dynamical system defined by ϕ is evolving on M. For additional details on Takens theorem, see [28], [29].

B. RCN Dynamics and Its Training Procedure

In this section, we introduce the dynamics of the RCN and a sufficient condition for the RCN to possess the ESP. With the guarantee of the ESP and using the Takens theorem, we show that an RCN can "learn" to forecast the data generated by a dynamical system on a compact manifold.

Consider the RCN described by

$$x[k+1] = (1-\alpha)x[k] + \alpha\sigma(Ax[k] + Bu[k]) \tag{1}$$

$$y[k] = Cx[k] = \sum_{i=1}^{N} c_i x_i[k] \tag{2}$$

where $x[k] \in \mathbb{R}^N$ denotes the state of the RCN, N is the number of nodes, $u[k] \in \mathbb{R}^p$ is the input to the RCN, $y[k] \in \mathbb{R}^q$ is the output of the RCN, $A \in \mathbb{R}^{N\times N}$ and $B \in \mathbb{R}^{N\times p}$ are predetermined matrices denoting the connections in the RCN, $C \in \mathbb{R}^{q\times N}$ is the weight matrix in the output layer to be trained, with $c_i, i = 1, \ldots, N$ as its ith column, $\alpha \in (0, 1)$ is called the "leakage rate," and σ is an activation function that is applied to a vector componentwise. A well-known result (see [1], [30]) for the RCN in (1) to possess ESP is provided as the following lemma.

Lemma 1: Consider the dynamics of the RCN in (1). If σ is Lipschitz continuous with the Lipschitz constant L, then the ESP holds if $\|A\|_2 < (1/L)$, where $\|\cdot\|_2$ denotes the matrix 2-norm.

Now, we illustrate the training process of the RCN. Given a time series $u[0; T-1]$ for $T > 1$ and a reference sequence $\tilde{y}[1; T]$, the RCN can be trained to predict the value of $\{\tilde{y}[k]\}$ for $k > T$. The canonical way to do this is to first select the size of the RCN (i.e., N) and randomly generate the matrices A and B of appropriate dimensions. Then, the sequence $u[0; T-1]$ is fed into the RCN dynamics (1) as an input to generate a sequence of the RCN states $x[1; T]$. A fixed positive integer w is selected as the "washout" length, and only the sequence after the wth step, i.e., $x[w; T]$, is collected. Finally, the coefficient in the output layer, C, is trained to minimize the error between the RCN output $y[w; T]$ and the reference sequence $\tilde{y}[w; T]$, i.e.,

$$C = \underset{C\in\mathbb{R}^{q\times N}}{\operatorname{argmin}} \sum_{k=w}^{T} \left\| C\begin{bmatrix} x[k] \\ u[k] \end{bmatrix} - \tilde{y}[k] \right\|_2^2.$$

Following this supervised learning procedure, when the input to the RCN is $u[k]$ (for $k > T$), the RCN outputs a value that approximates $\tilde{y}[k+1]$. In this sense, the training process enables the RCN to learn the underlying dynamics of the reference sequence $\tilde{y}[k]$.

In Section III-C, we explain in detail on how the RCN encodes the underlying dynamics of the reference sequence during the training process through the lens of Takens embedding theorem.

C. Learning Dynamics Using an RCN

To begin with, we consider the task of one-step ahead forecast of a given time series and explain how the training process enables the RCN to perform this task. For ease of exposition, we illustrate the idea with 1-D time series, i.e., $u[k], y[k] \in \mathbb{R}$, and the framework is directly applicable to the multidimensional cases since the Takens theorem holds regardless of the dimension of the time series.

1) One-Step Ahead Forecast: Suppose that a time series $\{u[k]\} \subset \mathbb{R}$ is generated by a dynamical system on a compact manifold of dimension m. The sequence $u[0; T]$ is used as input to the RCN, and the one-step shifted sequence $u[1; T+1]$ is provided as the training reference. Denote the solution for the state equation in (1) as $x_i[k] = \varphi_i(x[0], u[0], u[1], \ldots, u[k-1])$, where $x_i[k]$ is the ith component of the vector $x[k]$, and then, by the uniqueness of the solution of a dynamical system, we have $x_i[k] = \varphi_i(x[0], u[0], u[1], \ldots, u[k-1]) = \varphi_i(x[j], u[j], u[j+1], \ldots, u[k-1])$ for any $j = 0, 1, 2, \ldots, k-1$.

As a result of Lemma 1, when $\|A\|_2 < (1/L)$, the RCN acquires the ESP. This implies that there exists $k_0 \in \mathbb{N}$ such that, after the RCN evolves for k_0 steps, the effect of the initial condition on the solution trajectory becomes negligible. Therefore, for a fixed "washout" length $w > k_0$, we can define $\xi_{(i,w)} : \mathbb{R}^w \to \mathbb{R}^N$ such that

$$\begin{aligned} x_i[w+j] &= \varphi_i(x[j], u[j], \ldots, u[w+j-1]) \\ &:= \xi_{(i,w)}(u[j], \ldots, u[w+j-1]) \end{aligned}$$

for all $j = 0, \ldots, T-w$, where $\xi_{(i,w)}$ can be treated as φ_i taking historical data of length w, with the effect of the initial condition washed out. Recall that when training the RCN, we minimize the error between $y[w; T]$ and $\tilde{y}[w; T]$. Hence, training the RCN is equivalent to finding c_i's such that

$$\begin{aligned} u[w+j] \approx y[w+j] &= \sum_{i=1}^{N} c_i x_i[w+j] \\ &= \sum_{i=1}^{N} c_i \varphi_i(x[j], u[j], u[j+1], \ldots, u[w+j-1]) \\ &= \sum_{i=1}^{N} c_i \xi_{(i,w)}(u[j], u[j+1], \ldots, u[w+j-1]) \end{aligned} \tag{3}$$

for $j = 0, \ldots, T-w$.

We observe from (3) that when the RCN is endowed with the ESP, the training procedure is equivalent to learning the map between $u[w+j]$ and the window of historical data $u[j; w+j-1]$. The existence of such a map is guaranteed by the Takens theorem. In particular, with w in (3) greater than $2m+1$, there exists a smooth map $\psi_w : \mathbb{R}^w \to \mathbb{R}^w$ such that $\psi_w(u[j; w+j-1]) = u[j+1; w+j]$. Therefore, training an RCN can also be interpreted as approximating the map ψ_w using nonlinear functions $\xi_{(i,w)}$ for $i = 1, \ldots, N$.

2) Multistep Ahead Forecasting: Based on the idea of one-step ahead forecast, we can extend the RCN to accomplish τ-step ahead forecast. Specifically, if the training reference is set to be $\tilde{y}[w; T] = u[w+\tau-1; T+\tau-1]$, then similar to (3), training the RCN is equivalent to finding c_i's such that $u[w+j+\tau-1] \approx \sum_{i=1}^{N} c_i \xi_{(i,w)}(u[j; w+j-1])$ for $j = 0, \ldots, T-w$. A similar argument to one-step ahead forecasting holds for τ-step forecasting since the output of the RCN can be configured to approximate $\psi_w^\tau = \psi_w \circ \cdots \circ \psi_w$ (τ times)

such that $\psi_w^\tau(u[j; w+j-1]) = u[j+\tau; w+j+\tau-1]$. Hence, in this way, the RCN training can be viewed as approximating the last component of ψ_w^τ by using $\sum_{i=1}^N c_i \xi_{(i,w)}$.

Remark 1: The above explanation on multistep ahead forecasting based on time-delay embedding provides a much better understanding on why the RCN successfully forecasts a chaotic system as reported in [31] and [32]. In particular, for a chaotic system with an attractor, such as the Lorenz system, the sequence $\{u[k]\}$ will eventually lie in a compact manifold of low dimension. Hence, the Takens theorem together with the analysis presented above on time-delay embedding can be directly applied.

3) Learning Dynamics of Observation Sequences: In addition to forecasting the input sequence $u[k]$ for one-step/multistep ahead, where $\tilde{y}[k] = u[k]$, we can also configure RCN to forecast observation sequences that can be expressed as functions of $u[k]$. In particular, if $\tilde{y} = h(u)$, where h is a smooth function, then $\tilde{y}[k]$ can be represented solely by the history of $u[k]$, say $\tilde{y}[w+j] = G(u[j; w+j-1])$ since $u[k]$ is uniquely determined by a window of historical data. Therefore, the training process of the RCN, in this case, can be interpreted as using $\sum_{i=1}^N c_i \xi_{(i,w)}$ to approximate the nonlinear mapping G [similar to (3)]. This illustrates the ability of the RCN to forecast a wide variety of time series by simply changing the training reference $\tilde{y}[k]$.

4) Choices of the Activation Function: When the activation function σ is Lipschitz continuous with the Lipschitz constant L, the condition $\|A\|_2 < (1/L)$ is sufficient to ensure the ESP for (1) as a result of Lemma 1. For instance, the commonly used hyperbolic tangent function, tanh, has a Lipschitz constant $L = 1$ so that we need $\|A\|_2 < 1$ to ensure the ESP for (1). Since the above explanation on how RCNs learn dynamical systems does not post any restriction on the activation function σ, we indeed have much freedom on the choice of σ. In fact, the activation function can be as simple as a linear function.

The tradeoff between the RCN performance and the choice of activation function for the RCN is well-documented. Although it is believed, in general, that nonlinear activation functions perform better than linear ones, it is reported in the literature (see [33]–[36]) that this conclusion is indeed based on a case-by-case study. One major advantage of using linear activation functions, as we shall see in Section IV, is that a linear activation function enables a thorough analysis of the RCN using system-theoretic tools and allows for an explicit design of the RCN, which can eventually be used as a baseline for designing an RCN with a nonlinear activation function, e.g., tanh or sigmoid, as widely used in the literature.

IV. Realization Theory for the RCN Design

In this section, we present a realization-theoretic framework for systematic design of RCNs. We illustrate the main idea and conduct the analysis for RCNs with linear activation functions. Specifically, we first show that for given input–output sequences $u[0; T-1] \subset \mathbb{R}^p$ and $\tilde{y}[1; T] \subset \mathbb{R}^q$, there exists an RCN that approximates the input–output relation in the data. Then, we provide a systematic scheme to prune the size of RCNs while maintaining the same training error. At the end of this section, we illustrate how these results can be carried over to the design of RCNs with general nonlinear activation.

A. Realization of RCNs With Linear Activation Function

We begin by introducing the notion of realization from systems theory [37] and defining it in the context of the RCN, which will form the basis of the proposed RCN design framework.

Definition 1 (Realization of Linear Systems): Given two sequences $u[0; T-1] \subset \mathbb{R}^p$ and $\tilde{y}[1; T] \subset \mathbb{R}^q$, we say that the triplet, $\tilde{A} \in \mathbb{R}^{N\times N}$, $\tilde{B} \in \mathbb{R}^{N\times p}$, and $\tilde{C} \in \mathbb{R}^{q\times N}$, is an N-dimensional ϵ-error realization of the pair $(u[0; T-1], \tilde{y}[1; T])$ if the linear system

$$\begin{aligned} x[k+1] &= \tilde{A}x[k] + \tilde{B}u[k] \\ y[k] &= \tilde{C}x[k] \end{aligned} \tag{4}$$

satisfies $(\sum_{k=1}^T \|y[k] - \tilde{y}[k]\|_2^2)^{1/2} \le \epsilon$. Furthermore, if $\epsilon = 0$, then such a realization is called an N-dimensional perfect realization.

For simplicity, we will refer to an N-dimensional realization using $(\tilde{A}, \tilde{B}, \tilde{C})_N$, and this will denote the linear dynamical system in (4). To facilitate the delineation between two realizations, we introduce Markov parameters, equivalent and irreducible realizations as follows.

Definition 2 (Markov Parameter): The kth Markov parameter of a realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$ is a matrix of real numbers $\gamma_k \in \mathbb{R}^{q\times p}$ defined by $\gamma_k = \tilde{C}\tilde{A}^k\tilde{B}$.

Definition 3 (Equivalent Realizations): Two realizations $(A_1, B_1, C_1)_{N_1}$ and $(A_2, B_2, C_2)_{N_2}$ are said to be *equivalent* if $\gamma_k^{(1)} = \gamma_k^{(2)}$ holds for all $k = 0, 1, 2, \ldots$, where $\gamma_k^{(1)} = C_1 A_1^k B_1$ and $\gamma_k^{(2)} = C_2 A_2^k B_2$.

Definition 4 (Irreducible Realization): A realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$ is said to be irreducible if there exists no equivalent realization $(\hat{A}, \hat{B}, \hat{C})_{\bar{N}}$ with $\bar{N} < N$.

Remark 2: In the literature of control systems, Definition 4 is referred as the "minimal realization" if $(\tilde{A}, \tilde{B}, \tilde{C})$ is perfect [37]. Since, in this work, we consider the reduction of ϵ-error realizations of RCNs, Definition 4 is named "irreducible realization" to avoid ambiguity.

Next, we will describe the RCN training procedure through the use of a realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$ and then establish the realization framework tailored for the RCN that explicitly accounts for the ESP, an important and necessary property for the functioning of the RCN.

Consider the RCN as given in (1) with a linear activation function, e.g., σ is the identity function, given by

$$x[k+1] = [(1-\alpha)I + \alpha A]x[k] + \alpha B u[k]$$

where I is the identity matrix of appropriate dimension. If $\tilde{A} := (1-\alpha)I + \alpha A \in \mathbb{R}^{N\times N}$ and $\tilde{B} = \alpha B \in \mathbb{R}^{N\times p}$, then the RCN dynamics can be expressed as

$$x[k+1] = \tilde{A}x[k] + \tilde{B}u[k]. \tag{5}$$

Now, we introduce the connection between the RCN training procedure and the notion of realization theory. Specifically, the first step of RCN training is to fix the dimension N, the leakage rate α, and the randomly generate matrices $\tilde{A}$ and $\tilde{B}$.

Then, training the output layer of the RCN constructed using $\tilde{A}$ and $\tilde{B}$ is equivalent to finding $\tilde{C}^*$ such that

$$\tilde{C}^* = \underset{\tilde{C}\in\mathbb{R}^{q\times N}}{\operatorname{argmin}} E\left[\left(\tilde{A}, \tilde{B}, \tilde{C}\right)_N\right] \tag{6}$$

where $E[(\tilde{A}, \tilde{B}, \tilde{C})_N] := (\sum_{k=1}^{T} \|\tilde{y}[k] - \tilde{C}\tilde{A}^k\tilde{B}u[k]\|_2^2)^{1/2}$ denotes the training error of the realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$.

Let us consider an N_1-dimensional ϵ-error realization, denoted $(\tilde{A}_1, \tilde{B}_1, \tilde{C}_1)_{N_1}$, and suppose that $(\tilde{A}_2, \tilde{B}_2, \tilde{C}_2)_{N_2}$ is the irreducible equivalent realization of $(\tilde{A}_1, \tilde{B}_1, \tilde{C}_1)_{N_1}$ with $N_2 \leq N_1$. Then, it holds that

$$\begin{aligned}\min_{\tilde{C}\in\mathbb{R}^{q\times N_2}} E\left[\left(\tilde{A}_2, \tilde{B}_2, \tilde{C}\right)_{N_2}\right] &\leq E\left[\left(\tilde{A}_2, \tilde{B}_2, \tilde{C}_2\right)_{N_2}\right] \\ &= E\left[\left(\tilde{A}_1, \tilde{B}_1, \tilde{C}_1\right)_{N_1}\right] < \epsilon\end{aligned}$$

which implies that if we train the RCN constructed using the N_2-dimensional matrices $\tilde{A}_2 \in \mathbb{R}^{N_2\times N_2}$ and $\tilde{B}_2 \in \mathbb{R}^{N_2\times p}$, the training error will be bounded above by ϵ. Therefore, finding the irreducible equivalent realization to a given RCN enables quantifying a smaller size of the RCN that provides a desired training accuracy.

To adopt this realization-theoretic idea for the RCN design, an additional constraint has to be imposed on the matrix A in order for the RCN to be equipped with the ESP. To achieve this, we propose the notion of α-stable realization.

Definition 5 (α-Stable Realization): Given $\alpha \in (0, 1]$, a realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$ is called an α-stable realization if $\|\tilde{A}\|_2 < \alpha$.

For instance, we know, by Lemma 1, that an RCN in (1) using a linear activation function σ with the Lipschitz constant $L = 1$ possesses the ESP when $\|A\|_2 < 1$. Therefore, in this case, for any $\alpha \in ((1/2), 1)$, having $\|\tilde{A}\|_2 < 2\alpha - 1$ in (5) is sufficient to guarantee the ESP. This is because

$$\begin{aligned}\|A\|_2 &= \left\|\frac{1}{\alpha}\left(\tilde{A} - (1-\alpha)I\right)\right\|_2 \leq \frac{1}{\alpha}\left[\left\|\tilde{A}\right\|_2 + (1-\alpha)\right] \\ &< \frac{1}{\alpha}[2\alpha - 1 + 1 - \alpha] = 1.\end{aligned}$$

As a consequence, finding a $(2\alpha - 1)$-stable realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$ will ensure that the corresponding RCN possesses the ESP.

Therefore, in the remainder of this section, we will consider a fixed leakage rate $\alpha \in ((1/2), 1)$ so that $2\alpha - 1 \in (0, 1)$. For simplicity, we use "stable realization" in place of '$(2\alpha - 1)$-stable realization.'

B. Irreducible Stable Realization

Theoretically, if there exists one stable realization that achieves the desired training error for a given input–output sequence, one can construct many different realizations with the same training error. A consequent question of paramount practical importance to ask is how to prune the size of a realization as much as possible while maintaining the training error tolerance of the given stable realization. The answer to this question is pertinent to the concept of fundamental properties of a control system.

Definition 6 (Controllability and Observability Matrices): For a linear time-invariant dynamical system $(\tilde{A}, \tilde{B}, \tilde{C})_N$ as modeled in (4), the controllability and the observability matrices are defined by

$$W_N = \left[\tilde{B}, \tilde{A}\tilde{B}, \ldots, \tilde{A}^{N-1}\tilde{B}\right] \text{ and } G_N = \begin{bmatrix}\tilde{C} \\ \tilde{C}\tilde{A} \\ \vdots \\ \tilde{C}\tilde{A}^{N-1}\end{bmatrix}$$

respectively.

Controllability and observability properties then lead to the characterization of an irreducible realization (see [38]).

Lemma 2: A realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$ is irreducible if and only if the pair $(\tilde{A}, \tilde{B})_N$ is controllable and the pair $(\tilde{A}, \tilde{C})_N$ is observable, i.e., rank $(W_N) = N$ and rank $(G_N) = N$.

Note that Lemma 2 poses no constraints on the matrix norm of the realization, and thus, an irreducible realization may be unstable. As a result, modifications have to be made in order to construct a stable irreducible realization resulting in an RCN with ESP. In the following, we develop a systematic scheme to construct an equivalent stable realization of RCN with reduced size.

Lemma 3: Given an N-dimensional α-stable realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$, if rank $W_N = \bar{N} < N$, then there exists an $\bar{N}$-dimensional α-stable realization $(\bar{A}, \bar{B}, \bar{C})_{\bar{N}}$ that is equivalent to $(\tilde{A}, \tilde{B}, \tilde{C})_N$.

Proof: Because rank $W_N = \bar{N} < N$, let $v_1, \ldots, v_{\bar{N}}$ be an orthonormal basis of $\mathcal{R}(W_N)$, the column space of W_N, and select $v_{\bar{N}+1}, \ldots, v_N$ such that $v_1, \ldots, v_N$ forms an orthonormal basis of $\mathbb{R}^N$. Also, we denote $V_1 = [v_1, \ldots, v_{\bar{N}}]$, $V_2 = [v_{\bar{N}+1}, \ldots, v_N]$, and $V = [V_1, V_2]$. Note that by construction, V is an orthonormal matrix so that $V^{-1} = V^\intercal$. Therefore, we have

$$V^{-1}\tilde{A}V = V^\intercal\tilde{A}V = \begin{bmatrix} V_1^\intercal\tilde{A}V_1 & V_1^\intercal\tilde{A}V_2 \\ V_2^\intercal\tilde{A}V_1 & V_2^\intercal\tilde{A}V_2\end{bmatrix}.$$

Note that each column of $\tilde{A}V_1$ lies in $\mathcal{R}(V_1)$. Since columns of V_2 are in the orthogonal complement of $\mathcal{R}(V_1)$ by construction, it holds that $V_2^\intercal\tilde{A}V_1 = 0$. Therefore, $V^{-1}\tilde{A}V$ can be rewritten as

$$V^{-1}\tilde{A}V = \begin{bmatrix} V_1^\intercal\tilde{A}V_1 & V_1^\intercal\tilde{A}V_2 \\ 0 & V_2^\intercal\tilde{A}V_2\end{bmatrix} := \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22}\end{bmatrix}. \tag{7}$$

Moreover, since every column of $\tilde{B}$ lies in $\mathcal{R}(W_N)$, we have $V_2^\intercal\tilde{B} = 0$ by the construction of V_2 so that

$$V^{-1}\tilde{B} = V^\intercal\tilde{B} = \begin{bmatrix} V_1^\intercal\tilde{B} \\ 0\end{bmatrix} := \begin{bmatrix} B_1 \\ 0\end{bmatrix}. \tag{8}$$

Now, we denote $\tilde{C}V = [C_1, C_2]$, where C_1 consists of the first $\bar{N}$ columns of $\tilde{C}V$ and C_2 is formed by the remaining $N - \bar{N}$ columns. Then, it can be observed that $(V^{-1}\tilde{A}V, V^{-1}\tilde{B}, \tilde{C}V)_N$ is equivalent to $(\tilde{A}, \tilde{B}, \tilde{C})_N$ since

$$\tilde{C}V\left(V^{-1}\tilde{A}V\right)^k V^{-1}\tilde{B} = \tilde{C}\tilde{A}^k\tilde{B}$$

holds for all $k = 0, 1, \ldots$ Furthermore, due to the structure provided by (7) and (8), it can be verified that

$$\begin{aligned}\tilde{C}V\left(V^{-1}\tilde{A}V\right)^k V^{-1}\tilde{B} &= [C_1, C_2]\left(\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22}\end{bmatrix}\right)^k \begin{bmatrix} B_1 \\ 0\end{bmatrix} \\ &= [C_1, C_2]\begin{bmatrix} A_{11}^k & * \\ 0 & A_{22}^k\end{bmatrix}\begin{bmatrix} B_1 \\ 0\end{bmatrix} = C_1 A_{11}^k B_1\end{aligned}$$

which implies that $(A_{11}, B_1, C_1)_{\bar{N}}$ is an $\bar{N}$-dimensional realization that is equivalent to $(\tilde{A}, \tilde{B}, \tilde{C})_N$. Now, it remains to show that $(A_{11}, B_1, C_1)_{\bar{N}}$ is α-stable, given that $(\tilde{A}, \tilde{B}, \tilde{C})_N$ is α-stable. By the definition of matrix 2-norm, we have

$$\|A_{11}\|_2 = \sup_{\substack{x\in\mathbb{R}^{\bar{N}} \\ \|x\|_2=1}} x^\intercal A_{11} x = \sup_{\substack{x\in\mathbb{R}^{\bar{N}} \\ \|x\|_2=1}} x^\intercal V_1^\intercal \tilde{A} V_1 x. \tag{9}$$

If $y = V_1 x$, then y is a vector in $\mathbb{R}^N$ satisfying

$$\begin{aligned}\|y\|_2^2 = y^\intercal y &= (x_1 v_1 + \cdots + x_{\bar{N}} v_{\bar{N}})^\intercal (x_1 v_1 + \cdots + x_{\bar{N}} v_{\bar{N}}) \\ &= x_1^2 + \cdots + x_{\bar{N}}^2 = \|x\|_2^2.\end{aligned}$$

Hence, (9) can be bounded by $\|A_{11}\|_2 = \sup_{\substack{y\in\mathbb{R}^N \\ \|y\|_2=1}} y^\intercal \tilde{A} y \leq \|\tilde{A}\|_2 < \alpha$, which concludes the proof. □

From a dual perspective, we also have the following lemma regarding the observability matrix.

Lemma 4: Given an α-stable realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$, if rank $G_N = \bar{N} < N$, then there exists an $\bar{N}$-dimensional α-stable realization $(\bar{A}, \bar{B}, \bar{C})_{\bar{N}}$ equivalent to $(\tilde{A}, \tilde{B}, \tilde{C})_N$.

The proof is omitted since it is similar to the proof of Lemma 3. With the help of Lemmas 3 and 4, we develop an explicit criterion on characterizing the irreducible stable realization for the RCN given in (5).

Theorem 2: A stable realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$ is irreducible, i.e., there exists no equivalent stable realization $(\bar{A}, \bar{B}, \bar{C})_{\bar{N}}$ with $\bar{N} < N$, if and only if rank $W_N G_N = N$.

Proof: We prove the theorem by proving the contraposition, i.e., $(\tilde{A}, \tilde{B}, \tilde{C})_N$ is not irreducible if and only if rank $W_N G_N < N$.

On the one hand, if $(\tilde{A}, \tilde{B}, \tilde{C})_N$ is not irreducible, then by Lemma 2, either rank $W_N < N$ or rank $G_N < N$. Therefore,

$$\text{rank}\,(W_N G_N) \leq \min(\text{rank}\,(W_N), \text{rank}\,G_N) < N.$$

On the other hand, if rank $(W_N G_N) < N$, then, by Sylvester's rank inequality, it holds that

$$\text{rank}\,(W_N) + \text{rank}\,(G_N) - N \leq \text{rank}\,(W_N G_N) < N$$

which implies that rank W_N + rank $G_N < 2N$ and that either rank $(W_N) < N$ or rank $(G_N) < N$. Hence, from Lemma 3 (or Lemma 4), it holds that $(\tilde{A}, \tilde{B}, \tilde{C})_N$ is not irreducible. □

As a consequence of Lemmas 3 and 4 and Theorem 2, the procedure for finding the irreducible stable realization that is equivalent to a given stable realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$ is described in Algorithm 1, where $\mathsf{Orth}(A)$ returns an orthonormal basis of $\mathcal{R}(A)$ and dim $\tilde{A}$ returns the dimension of the matrix $\tilde{A}$.

Remark 3: It is worthwhile to mention that as proved in [27], a linear RCN attains maximal memory capacity when its weight matrices (A, B) constitutes a full-rank controllability matrix. As a consequence, Algorithm 1 not only returns an irreducible linear realization of RCN but also provides an RCN that reaches maximum memory capacity.

Algorithm 1 Minimal Stable Realization

function MINIMAL STABLE REALIZATION($\tilde{A}, \tilde{B}, \tilde{C}$)
  Initialize: Compute W_N, G_N for $(\tilde{A}, \tilde{B}, \tilde{C})_N$
  while rank $(W_N G_N) < N$ **do**
    if rank $(W_N) < N$ **then**
      $V_1 = \mathsf{Orth}\,(W_N)$.
    else
      $V_1 = \mathsf{Orth}\,(G_N^\intercal)$.
    end if
    $\tilde{A} = V_1^\intercal \tilde{A} V_1$, $\tilde{B} = V_1^\intercal \tilde{B}$, $\tilde{C} = \tilde{C} V_1$,
    $N = \dim \tilde{A}$.
    Compute W_N, G_N for $(\tilde{A}, \tilde{B}, \tilde{C})_N$.
  end while
  return $(\tilde{A}, \tilde{B}, \tilde{C})_N$.
end function

Remark 4: Each iteration in Algorithm 1 consists of computing the eigendecomposition of the controllability or the observability matrix, which has a time complexity of $O(N^3)$. In the worst case, Algorithm 1 may take N iterations to terminate, which results in a total time complexity of $O(N^4)$. Nevertheless, in all our numerical experiments, we observe that the number of iterations for Algorithm 1 to terminate is of order much smaller than N, i.e., around 10–20 iterations for the cases when $N = 500$ and 1000 or even $N = 2000$ so that we empirically expect that the average time complexity of Algorithm 1 is $O(N^3)$. On the other hand, model selection procedures for designing an initial learning model for the given data typically involve evaluating the performance of models with various hyperparameters and choosing the model that yields the best result [20], [39]. One of the main features of the proposed approach is that the results of Theorem 2 can be used to evaluate the irreducibility of the RCN, and Algorithm 1 can be used to prune the RCN size, irrespective of how the initial RCN model is selected.

C. Further Implications of Irreducible Stable Realizations

The development of irreducible realization in Section IV-B has a lot to offer for designing linear and nonlinear RCNs, as well as understanding the underlying dynamics in the training dataset. We start by explaining how the size of the irreducible realization is related to the Takens embedding, which provides a criterion to characterize the complexity of the underlying dynamics determined by $u[0; T-1]$ and $\tilde{y}[1; T]$.

From the theory of linear dynamical system (see [40]), it is a known fact that for any N-dimensional realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$, there exists an invertible matrix $P \in \mathbb{R}^{N\times N}$ such that $(P^{-1}\tilde{A}P, P^{-1}\tilde{B}, \tilde{C}P)_N$ is in the observable canonical

form given by

$$P^{-1}\tilde{A}P = \begin{pmatrix} 0 & I_q & & & \\ 0 & 0 & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & 0 & \cdots & 0 & I_q \\ -\theta_0 I_q & -\theta_1 I_q & \cdots & \theta_{N-2} I_q & -\theta_{N-1} I_q \end{pmatrix}$$

$$P^{-1}\tilde{B} = \begin{pmatrix} \gamma_0 \\ \vdots \\ \gamma_{N-1} \end{pmatrix}, \quad \tilde{C}P = \begin{pmatrix} I_q, & 0, & \cdots, & 0 \end{pmatrix} \tag{10}$$

where θ_j, $j = 0, 1, \ldots, N-1$, are arbitrary constants and γ_j, $j = 0, 1, \ldots, N-1$, are the first N Markov parameters of the realization $(\tilde{A}, \tilde{B}, \tilde{C})_N$. It is not hard to verify that for any invertible matrix $P \in \mathbb{R}^{N\times N}$, we have $(P^{-1}\tilde{A}P, P^{-1}\tilde{B}, \tilde{C}P)_N$ to be equivalent to $(\tilde{A}, \tilde{B}, \tilde{C})_N$. Therefore, without loss of generality, we assume that $(\tilde{A}, \tilde{B}, \tilde{C})_N$ is in the observable canonical form as in (10). In this case, the dynamics of each component of $x[k]$ is given by

$$x[k+1] = \begin{pmatrix} x_2[k] + \gamma_0 u[k] \\ \vdots \\ x_N[k] + \gamma_{N-2} u[k] \\ \sum_{i=0}^{N-1} \theta_i x_i[k] + \gamma_{N-1} u[k] \end{pmatrix}, \quad \tilde{y}[k] = x_1[k]$$

where $x_i[k]$ is the ith component of $x[k]$. Therefore, we have

$$\begin{aligned} \tilde{y}[k+N] &= x_1[k+N] = x_2[k+N-1] + \gamma_0 u[k+N-1] \\ &= \cdots = x_N[k+1] + \sum_{i=0}^{N-1} \gamma_i u[k+N-1-i]. \end{aligned} \tag{11}$$

When the RCN possesses the ESP, the effect of $x_N[k+1]$ on $\tilde{y}[k+N]$ is negligible, so that from (11), $\tilde{y}[k+N]$ is determined by $u[k], \ldots, u[k+N-1]$, which is the input data history of size N. From the Takens theorem, the time evolution on a compact manifold of dimension m can be represented by any time-delay embedding longer than $2m+1$. Therefore, if a linear realization describes the underlying dynamics determined by $u[0; T-1]$ and $\tilde{y}[1; T]$ perfectly, the dimension of such realization satisfies $N > 2m+1$ by the Takens theorem. On the other hand, if a linear realization of size N attains a training error of ϵ, then it implies that there exists a dynamical system evolving on a manifold of dimension $(1/2)(N-1)$ that approximately represents the underlying dynamics up to ϵ-error. This analysis provides a bound on the size of an RCN representing the dynamics of the underlying dynamical system generating the given input–output data sequences.

In addition, it is worth mentioning that since we are using linear dynamics to approximate the map ψ_{2m+1} in the Takens embedding, the bound on m mentioned above can be improved through the use of a nonlinear realization. It is intuitive to argue that using a nonlinear realization (e.g., with tanh or sigmoid as the activation function) to design an RCN may result in better approximation compared to a linear realization of the same dimension. However, the explicit solution of

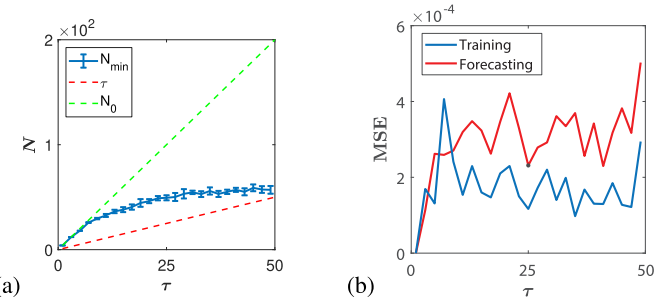


Fig. 1. Results of designing RCNs to forecast a time-delay system. (a) Size of irreducible 0.001-error realization versus length of time delay. (b) Training and forecasting MSE of the irreducible 0.001-error realization versus length of time delay.

RCN dynamics with a nonlinear activation function is in general unavailable. In this case, the linear realization of the RCN offers a guideline toward designing a nonlinear RCN realization. A reasonable design approach for the RCN with a nonlinear activation function is to first design a linear realization, say $(\tilde{A}, \tilde{B}, \tilde{C})_N$, for the given input–output data sequences; then, construct an RCN with nonlinear activation function using the matrices A and B as in (1) through

$$A = \frac{1}{\alpha}\big(\tilde{A} - (1-\alpha)I\big), \quad B = \frac{1}{\alpha}\tilde{B}$$

and train the readout layer again.

V. Numerical Experiments

In this section, we present several numerical examples to illustrate the developed tractable realization-theoretic approach with training error guarantees, for which the irreducible size of RCNs with respect to desired training errors can be explicitly quantified. Based on linear stable realizations, we further design nonlinear RCNs with the canonical tanh or sigmoid activation functions that achieve the desired training error performance.

A. Irreducible Realization of Time-Delay Systems

In this example, we design an RCN using the proposed approach for forecasting a time-delay system to elucidate the intimate connection between the irreducible linear realization and the Takens embedding.

We first introduce how the training data were generated and how the RCN was trained. For a fixed time delay τ, we randomly picked $u[0], \ldots, u[\tau-1]$ under a uniform distribution on $[-1, 1]$. Then, we completed the sequence of u by setting $u[k+\tau] = u[k]$ for $k = 0, \ldots, T-1$. In this way, the dynamical system governing the sequence $u[0; T]$ is a τ-step time-delay system. After generating the sequence $u[0; T]$, we used $u[0; T-\tau]$ as the input and $u[\tau, T]$ as the reference output to train an RCN with linear activation function.

In this experiment, we varied τ from 1 to 50. For each τ, we randomly generated an N_0-dimensional 0.001-error realization with $N_0 > \tau$ and computed its irreducible realization using Algorithm 1. Then, using the irreducible realization,

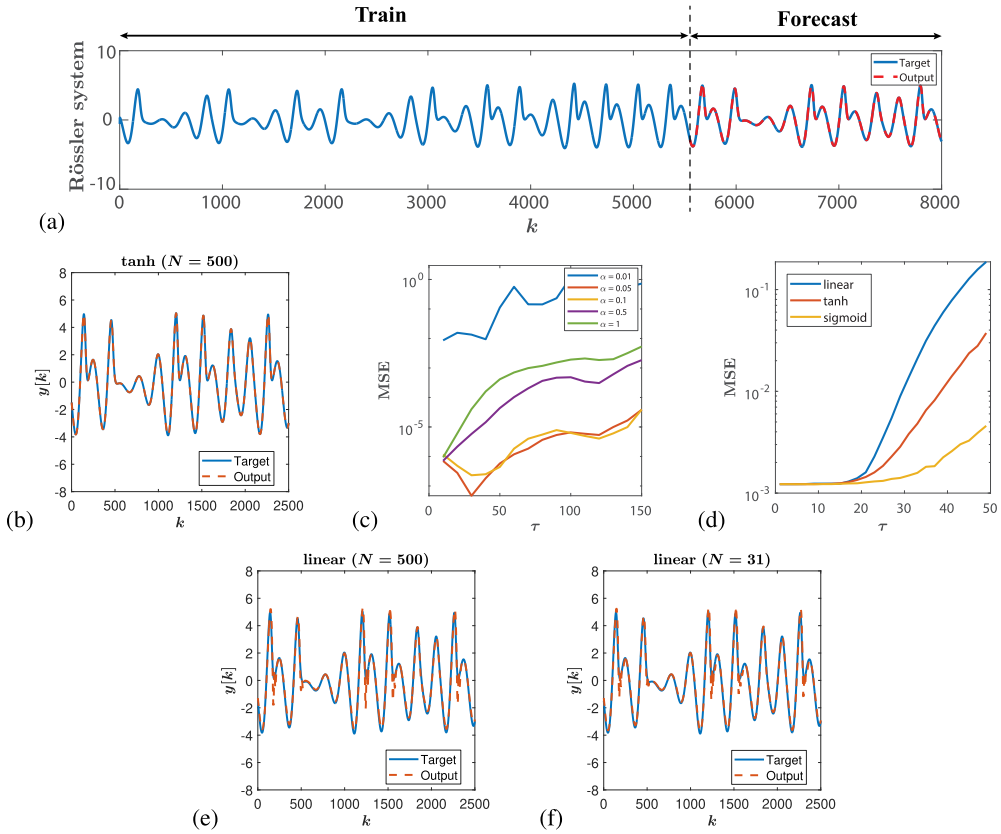


Fig. 2. (a) Experiment setup for using RCN to predict the time series generated by the Rössler system for 30 steps ahead. (b) Forecasting the same time series as (a) using the tangent hyperbolic function. (c) MSE between the forecast output and target, versus different time delays, using the same RCN as in (b) with different leakage rates. (d) MSE between the forecast output and target versus different time delays using the same RCN as in (b) under various activation functions. (e) Forecasting the same times series as (a), but changing the activation function into a linear function. (f) Forecasting the same time series as (a) using the reduced RCN computed from (e) through Algorithm 1.

we forecast the sequence τ-step ahead for 2000 time steps. When training RCNs, we fixed training length as $T = 1000$ and leakage rate as $\alpha = 0.9$.

Fig. 1(a) shows the size of irreducible linear realization $N_{\min}$ with respect to τ with N_0 selected as $N_0 = 4\tau$. Each point in the figure is the average plus/minus the standard deviation of 10 independent experiments under the same setup. Fig. 1(a) shows that using Algorithm 1, we can trim a large RCN (green dashed line) into much smaller size (blue solid line) with the same performance. This result is consistent with our analysis in Section IV-C that the minimum size of an RCN realization can be used as a criterion to quantify the length of time-delay embedding (red dashed line) associated with the training dataset. Fig. 1(b) shows the average mean squared errors (MSEs) of the irreducible linear realization with respect to τ.

B. Time-Evolution Forecast for Chaotic Systems

In this example, we show the use of an RCN to learn the temporal evolution of a chaotic system and analyze how the network configuration affects the performance of the designed RCN. Specifically, we consider the Rössler system, which is given by

$$dx/dt = -y - z, \quad dy/dt = x + ay, \quad dz/dt = b + z(x - c)$$

where x, y, and z are state variables and a, b, and c are constant parameters selected as $a = 0.5$, $b = 2.0$, and $c = 4.0$. Initial conditions were selected as $x(0) = 0$, $y(0) = 0$, and $z(0) = 1$.

The Rössler system was simulated for $t \in [0, 1]$, with 8000 sampling points collected in this time window. The first 5000 points were used as the training input, and the sample points between 30 and 5030 were used as the reference sequence to train the RCN for 30-step ahead forecasting. Then, we recorded the output of the RCN for another 3000 steps as a forecasting sequence. Fig. 2(a) shows a demonstration of forecasting the x-component of Rössler system for 30-steps ahead.

We first randomly generate two RCNs with all hyperparameters to be the same, but one with tanh activation function and the other one with linear activation function. The hyperparameters of the RCN were fixed as follows: number of nodes $N = 500$, leakage rate $\alpha = 0.8$, length of training data $t_r = 5000$, length of forecasting data $t_s = 3000$, and washout length $w = 500$. If not specifically mentioned, the matrix A was generated under a uniform distribution on [0, 1] of sparsity 63.2% and then normalized to have a matrix 2-norm equals to 0.9. The output layer was trained via ridge regression [41] with $\lambda = 10^{-8}$.

Fig. 2(b) presents the results of forecasting the time series provided in Fig. 2(a) with an MSE of 1.49×10^{-3}. Under the

TABLE I

RESULTS OF FORECASTING TIME-DELAY SYSTEM USING LINEAR AND NONLINEAR RCNS. IN EACH EXPERIMENT, WE CONSTRUCT A LINEAR RCN AND TWO NONLINEAR RCNS (WITH tanh AND sigmoid ACTIVATION FUNCTIONS) OF SIZE N USING THE SAME HYPERPARAMETERS AND THE SAME RANDOMLY GENERATED $\tilde{A}$ AND $\tilde{B}$. THE ERROR OF USING RCNS OF SIZE N TO FORECAST AN N-STEPS TIME-DELAY SYSTEM IS PROVIDED IN THE TABLE, WHERE 2000 INDEPENDENT EXPERIMENTS ARE CONDUCTED FOR EACH N

N	Training MSE (mean ± std)			Forecasting MSE (mean ± std)			$\mathbb{P}(\epsilon_{\text{linear}} > \max(\epsilon_{\tanh}, \epsilon_{\text{sigmoid}}))$
	linear ($\times 10^{-5}$)	tanh ($\times 10^{-5}$)	sigmoid ($\times 10^{-7}$)	linear ($\times 10^{-5}$)	tanh ($\times 10^{-5}$)	sigmoid ($\times 10^{-7}$)	
50	0.11 ± 0.00	0.10 ± 0.00	1.55 ± 0.50	0.28 ± 0.00	0.24 ± 0.02	2.53 ± 0.69	99.85%
100	0.04 ± 0.00	0.04 ± 0.00	0.53 ± 0.13	0.11 ± 0.00	0.11 ± 0.00	1.06 ± 0.25	100%
150	3.78 ± 0.00	3.62 ± 0.04	0.32 ± 0.06	1.83 ± 0.00	1.67 ± 0.03	0.65 ± 0.14	100%
200	4.63 ± 0.00	4.52 ± 0.02	0.24 ± 0.04	11.57 ± 0.00	11.07 ± 0.00	0.47 ± 0.08	100%
250	0.89 ± 0.00	0.86 ± 0.00	0.20 ± 0.03	2.23 ± 0.00	2.07 ± 0.02	0.38 ± 0.06	100%
300	2.00 ± 0.00	1.97 ± 0.00	0.18 ± 0.02	2.38 ± 0.00	2.37 ± 0.00	0.32 ± 0.04	100%
350	0.02 ± 0.00	0.02 ± 0.00	0.16 ± 0.02	0.53 ± 0.00	0.52 ± 0.00	0.27 ± 0.03	99.95%
400	2.12 ± 0.00	2.11 ± 0.00	0.15 ± 0.01	1.40 ± 0.00	1.37 ± 0.00	0.25 ± 0.03	99.85%
450	4.60 ± 0.00	4.59 ± 0.00	0.14 ± 0.01	9.83 ± 0.00	9.41 ± 0.00	0.23 ± 0.02	100%
500	4.45 ± 0.00	4.36 ± 0.00	0.13 ± 0.01	11.11 ± 0.00	10.68 ± 0.03	0.21 ± 0.02	100%

TABLE II

RESULTS OF FORECASTING RÖSSLER SYSTEM USING LINEAR AND NONLINEAR RCNS. IN EACH EXPERIMENT, WE CONSTRUCT A LINEAR RCN AND TWO NONLINEAR RCNS (WITH tanh AND sigmoid ACTIVATION FUNCTIONS) OF SIZE N USING THE SAME HYPERPARAMETERS AND THE SAME RANDOMLY GENERATED $\tilde{A}$ AND $\tilde{B}$. THE ERROR OF USING RCN OF DIFFERENT SIZES TO FORECAST THE RÖSSLER SYSTEM IN SECTION V-B FOR TEN STEPS AHEAD IS PROVIDED IN THE TABLE, WHERE 2000 INDEPENDENT EXPERIMENTS ARE CONDUCTED FOR EACH N

N	Training MSE (mean ± std)			Forecasting MSE (mean ± std)			$\mathbb{P}(\epsilon_{\text{linear}} > \max(\epsilon_{\tanh}, \epsilon_{\text{sigmoid}}))$
	linear ($\times 10^{-3}$)	tanh ($\times 10^{-5}$)	sigmoid ($\times 10^{-5}$)	linear ($\times 10^{-3}$)	tanh ($\times 10^{-5}$)	sigmoid ($\times 10^{-5}$)	
50	2.96 ± 0.08	8.12 ± 2.16	5.09 ± 0.90	7.33 ± 0.06	21.6 ± 8.18	5.68 ± 1.18	100%
100	2.88 ± 0.06	3.27 ± 0.46	3.72 ± 0.29	7.39 ± 0.05	4.94 ± 1.61	3.81 ± 0.33	
150	2.85 ± 0.05	2.33 ± 0.29	3.36 ± 0.20	7.42 ± 0.04	2.83 ± 0.63	0.34 ± 0.23	
200	2.83 ± 0.04	1.86 ± 0.24	3.15 ± 0.15	7.44 ± 0.03	2.17 ± 0.35	0.32 ± 0.18	
250	2.81 ± 0.04	1.55 ± 0.19	2.99 ± 0.14	7.45 ± 0.03	1.83 ± 0.24	3.07 ± 0.16	
300	2.80 ± 0.03	1.34 ± 0.17	2.86 ± 0.12	7.46 ± 0.03	1.64 ± 0.19	2.93 ± 0.14	
350	2.79 ± 0.03	1.18 ± 0.15	2.75 ± 0.10	7.47 ± 0.03	1.49 ± 0.15	2.82 ± 0.12	
400	2.78 ± 0.03	1.05 ± 0.14	2.65 ± 0.09	7.48 ± 0.02	1.39 ± 0.13	2.71 ± 0.12	
450	2.77 ± 0.03	0.96 ± 0.13	2.57 ± 0.09	7.49 ± 0.02	1.31 ± 0.12	2.62 ± 0.11	
500	2.76 ± 0.02	0.88 ± 0.12	2.48 ± 0.09	7.49 ± 0.02	1.23 ± 0.10	2.53 ± 0.10	

same setup as in Fig. 2(b), we varied the leakage rate α and the activation function of the RCN and present the corresponding results of MSE versus time delay (τ) in Fig. 2(c) and (d), respectively. From Fig. 2(c), we observed that $\alpha = 0.05$ resulted in the best MSE across different cases (of τ), and from Fig. 2(d), we observed that the RCN with linear activation function achieved a similar performance as the RCNs with a nonlinear activation function when the time delay τ was small. Therefore, we used the same RCN as in Fig. 2(b) but changed the leakage rate to $\alpha = 0.05$ and the activation function into linear function to forecast the same time series. The corresponding results are presented in Fig. 2(e) with an MSE of 2.97×10^{-2}.

We further applied Algorithm 1 on the above RCN with linear activation function, yielding an irreducible linear realization of size 31 with an MSE of 2.54×10^{-2}. Fig. 2(f) presents the result of using the irreducible RCN to forecast the same time series generated by the Rössler system for 30-steps ahead. Based on our empirical analysis in Section IV-C, the results in Fig. 2(f) implies that the underlying dynamics of the given input–output sequences can be well-approximated by a linear dynamics with a time delay of no longer than 31 steps, which is evident by the experiment setups.

C. Analysis of Linear and Nonlinear Activation Functions

In this section, we further investigate the difference in performance between linear and nonlinear RCNs. The results in this section support our idea of using the linear realization of the RCN to help design a nonlinear RCN with a guaranteed training error, as mentioned in Section IV-C.

Table I presents the results of forecasting time-delay systems using linear and nonlinear RCNs. We varied the size of the RCN from 50 to 500. For each N, we generated a time-delay system with N delay steps (as in Section V-A) and simulated 2000 independent experiments of forecasting $(N-1)$-steps ahead. The training length was fixed as $t_r = 5000$ and the forecast length was fixed as $t_s = 2000$. To make a fair comparison, in each experiment, we generated three RCNs of size N, one with linear activation function, one with tanh activation function, and another with sigmoid activation function, using the same randomly generated matrices $\tilde{A}$ and $\tilde{B}$. Other hyperparameters of the three RCNs were set to be the same as in Section V-A. The average and standard deviation of training and forecast MSE are reported in Table I. As we observe from the last column in Table I, the nonlinear RCNs outperform the linear RCN in terms of the training MSE in most cases.

Table II reports the results of forecasting the Rössler system using RCNs with different activation functions. Using the same dataset as the one in V-B, we compared the performance of linear and nonlinear RCN at different scales. Similar to the previous table, we varied the size of the RCN from 50 to 500 and conducted 2000 independent experiments to forecast the Rössler system for ten-steps ahead. The training

length was fixed as $t_r = 5000$ and the forecast length was fixed as $t_s = 2000$. In each experiment, we generated three RCNs of size N, one with linear activation function, one with tanh activation function, and another with sigmoid activation function, using the same randomly generated matrices $\tilde{A}$ and $\tilde{B}$. Other hyperparameters of the three RCNs were set to be the same as in Section V-B. The average and standard deviation of training and forecast MSE are reported in Table II. As we observe from the last column in Table II, in this experiment, the training MSE of nonlinear RCNs was always smaller than that of the linear RCNs.

VI. Conclusion

In this article, we provided a detailed analysis and a holistic description of the training procedure and the operation of the RCN. With the help of Takens embedding theorem, we derived the delay embedding map, which an RCN potentially learns during the training process from the given input–output data. This provided insights into the role that the linear activation function and other hyperparameters play in the design and working of RCNs in applications such as forecasting a time series. Furthermore, based on the notions of linear realization theory, we provided a systematic approach to trim RCNs with guaranteed training accuracy. In this context, we introduced the idea of α-stable realizations for designing stable RCNs that achieve the desired training objective with reduced size and established a tractable design algorithm to synthesize RCNs with nonlinear activation. The numerical experiments on forecasting time-delay systems and the Rössler system were used to substantiate our proposed design approach for interpretable RCNs. We observed from the experiments that the nonlinear RCNs with both reduced size and guaranteed training accuracy can be attained based on the minimum realization for linear RCNs. These results suggested that the proposed approach offers an informed and interpretable design methodology to devise nonlinear RCNs for a given dataset to decode the underlying dynamics in the data.

References

[1] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note," German Nat. Res. Center Inf. Technol., Bonn, Germany, Tech. Rep. 148, 2001, no. 34, p. 13.

[2] M. Lukoševičius, H. Jaeger, and B. Schrauwen, "Reservoir computing trends," *KI-Künstliche Intell.*, vol. 26, no. 4, pp. 365–371, Nov. 2012.

[3] G. Tanaka *et al.*, "Recent advances in physical reservoir computing: A review," *Neural Netw.*, vol. 115, pp. 100–123, Jul. 2019.

[4] X. Lin, Z. Yang, and Y. Song, "Short-term stock price prediction based on echo state networks," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 7313–7317, Apr. 2009.

[5] L. Grigoryeva, J. Henriques, L. Larger, and J.-P. Ortega, "Stochastic nonlinear time series forecasting using time-delay reservoir computers: Performance and universality," *Neural Netw.*, vol. 55, pp. 59–71, Jul. 2014.

[6] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, Apr. 2004.

[7] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, 2002.

[8] E. A. Antonelo, B. Schrauwen, and D. Stroobandt, "Event detection and localization for small mobile robots using reservoir computing," *Neural Netw.*, vol. 21, no. 6, pp. 862–871, Aug. 2008.

[9] M. Lukoševičius, *A Practical Guide to Applying Echo State Networks*. Berlin, Germany: Springer, 2012, pp. 659–686.

[10] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017, *arXiv:1702.08608*.

[11] Z. C. Lipton, "The mythos of model interpretability," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[12] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky-integrator neurons," *Neural Netw.*, vol. 20, no. 3, pp. 335–352, Apr. 2007.

[13] L. Grigoryeva and J.-P. Ortega, "Universal discrete-time reservoir computers with stochastic inputs and linear readouts using non-homogeneous state-affine systems," *J. Mach. Learn. Res.*, vol. 19, no. 1, pp. 892–931, Jan. 2018.

[14] M. Buehner and P. Young, "A tighter bound for the echo state property," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 820–824, May 2006.

[15] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," *Neural Netw.*, vol. 35, pp. 1–9, Nov. 2012.

[16] H. Jaeger, "Short term memory in echo state networks," German Nat. Res., Inst. Comput. Sci., GMD-Rep. 152.

[17] L. Grigoryeva, J. Henriques, L. Larger, and J.-P. Ortega, "Nonlinear memory capacity of parallel time-delay reservoir computers in the processing of multidimensional signals," *Neural Comput.*, vol. 28, no. 7, pp. 1411–1451, Jul. 2016.

[18] S. Marzen, "Difference between memory and prediction in linear recurrent networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 96, no. 3, Sep. 2017, Art. no. 032308.

[19] J. Boedecker, O. Obst, J. T. Lizier, N. M. Mayer, and M. Asada, "Information processing in echo state networks at the edge of chaos," *Theory Biosci.*, vol. 131, no. 3, pp. 205–213, 2012.

[20] A. Rodan and P. Tino, "Minimum complexity echo state network," *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 131–144, Jan. 2011.

[21] C. Gallicchio and A. Micheli, "Architectural and Markovian factors of echo state networks," *Neural Netw.*, vol. 24, no. 5, pp. 440–456, 2011.

[22] Y. Kawai, J. Park, and M. Asada, "A small-world topology enhances the echo state property and signal propagation in reservoir computing," *Neural Netw.*, vol. 112, pp. 15–23, Apr. 2019.

[23] B. Zhang, D. J. Miller, and Y. Wang, "Nonlinear system modeling with random matrices: Echo state networks revisited," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 1, pp. 175–182, Jan. 2012.

[24] L. Grigoryeva and J.-P. Ortega, "Dimension reduction in recurrent networks by canonicalization," 2020, *arXiv:2007.12141*.

[25] E. Bollt, "On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 31, no. 1, Jan. 2021, Art. no. 013108.

[26] P. Verzelli, C. Alippi, L. Livi, and P. Tino, "Input representation in recurrent neural networks dynamics," 2021, *arXiv:2003.10585*.

[27] L. Gonon, L. Grigoryeva, and J.-P. Ortega, "Memory and forecasting capacities of nonlinear recurrent networks," *Phys. D, Nonlinear Phenomena*, vol. 414, Dec. 2020, Art. no. 132721, doi: 10.1016/j.physd.2020.132721.

[28] J. Huke, "Embedding nonlinear dynamical systems: A guide to takens' theorem," Manchester Inst. Math. Sci., Univ. Manchester, Manchester, U.K., Tech. Rep. 84, 1993.

[29] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*. Berlin, Germany: Springer, 1981, pp. 366–381.

[30] L. Grigoryeva and J.-P. Ortega, "Differentiable reservoir computing," *J. Mach. Learn. Res.*, vol. 20, no. 179, pp. 1–62, 2019.

[31] P. Antonik, M. Gulina, J. Pauwels, and S. Massar, "Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronization and cryptography," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 98, no. 1, Jul. 2018, Art. no. 012215.

[32] T. L. Carroll, "Using reservoir computers to distinguish chaotic signals," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 98, no. 5, Nov. 2018, Art. no. 052209.

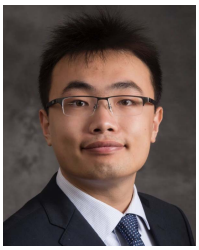
[33] D. Verstraeten, J. Dambre, X. Dutoit, and B. Schrauwen, "Memory versus non-linearity in reservoirs," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2010, pp. 1–8.

[34] P. Verzelli, C. Alippi, and L. Livi, "Echo state networks with self-normalizing activations on the hyper-sphere," *Sci. Rep.*, vol. 9, no. 1, pp. 1–14, Dec. 2019.

[35] M. Inubushi and K. Yoshimura, "Reservoir computing beyond memory-nonlinearity trade-off," *Sci. Rep.*, vol. 7, no. 1, p. 10199, Dec. 2017.

[36] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Netw.*, vol. 20, no. 3, pp. 391–403, Apr. 2007.

[37] B. D. Schutter, "Minimal state-space realization in linear system theory: An overview," *J. Comput. Appl. Math.*, vol. 121, nos. 1–2, pp. 331–354, Sep. 2000.
[38] L. Silverman, "Realization of linear dynamical systems," *IEEE Trans. Autom. Control*, vol. AC-16, no. 6, pp. 554–567, Dec. 1971.
[39] S. Aras and D. Kocakoç, "A new model selection strategy in time series forecasting with artificial neural networks: IHTS," *Neurocomputing*, vol. 174, pp. 974–987, Jan. 2016.
[40] R. W. Brockett, *Finite Dimensional Linear Systems*. Philadelphia, PA, USA: SIAM, 2015.
[41] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.



Wei Miao (Member, IEEE) received the B.Eng. degree from the Department of Automation, Tsinghua University, Beijing, China, in 2016, and the Ph.D. degree in systems science and mathematics from Washington University in St. Louis, St. Louis, MO, USA, in 2021.

He was a member of the McDonnell International Scholars Academy, Washington University in St. Louis. His research lies in fundamental developments in systems and control theory and their data-driven applications.

Vignesh Narayanan (Member, IEEE) received the B.Tech. degree from SASTRA University, Thanjavur, India, in 2012, the M.Tech. degree from the National Institute of Technology, Kurukshetra, India, in 2014, and the Ph.D. degree from the Missouri University of Science and Technology, Rolla, MO, USA, in 2017.

He was a Post-Doctoral Research Associate at Washington University in St. Louis, St. Louis, MO, USA. He is currently an Assistant Professor at the AI Institute and the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA. He is also with the UofSC Autism and Neurodevelopmental Disorders Center of Excellence. His research interests include dynamical systems and networks, data science, learning, and computational neuroscience.

Jr-Shin Li (Senior Member, IEEE) received the B.S. and M.S. degrees from National Taiwan University, Taipei, Taiwan, in 1996 and 1998, respectively, and the Ph.D. degree in applied mathematics from Harvard University, Cambridge, MA, USA, in 2006.

He is currently Sarah Louisa Glasgow and Newton Wilson Professor in Electrical and Systems Engineering with a joint appointment at the Division of Biology and Biomedical Sciences, Washington University in St. Louis, St. Louis, MO, USA. His research interests are in the areas of control theory, computational mathematics, optimization, learning, data science, and complex networks. His current work involves developing model-based and data-driven methods for dynamical systems and control of large-scale and ensemble systems with applications from neuroscience and biology to quantum physics.

Dr. Li was a recipient of the NSF CAREER Award in 2008 and the AFOSR Young Investigator Award in 2010.