Interpretable Design of Reservoir Computing Networks Using Realization Theory

Wei Miao
LinkedIn

Vignesh Narayanan
University of South Carolina - Columbia

Jr-Shin Li
Washington University in St. Louis

Follow this and additional works at: https://scholarcommons.sc.edu/aii_fac_pub

Part of the Applied Mathematics Commons, Artificial Intelligence and Robotics Commons, and the Theory and Algorithms Commons

Publication Info
© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

This Article is brought to you by the Artificial Intelligence Institute at Scholar Commons. It has been accepted for inclusion in Publications by an authorized administrator of Scholar Commons. For more information, please contact digres@mailbox.sc.edu.
Interpretable Design of Reservoir Computing Networks using Realization Theory

Wei Miao, Student 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 paper, 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 number of hidden nodes in linear RCNs based on the concepts of controllability and observability matrices. Leveraging the linear RCN design, we provide a tractable procedure to realize RCNs with nonlinear activation functions. Finally, 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—Reservoir computing network (RCN), Realization theory, Time-series forecasting

I. INTRODUCTION

The reservoir computing network (RCN) is a bio-mimetic 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 pre-determined 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 empirical design of the hyper-parameters 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 hyper-parameters that yield a desired performance, is compelling but challenging. In this paper, 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 matrices, we derive a necessary and sufficient condition on the irreducibility of RCNs with linear activation functions. This in turn results in an interpretable RCN pruning algorithm, where the RCNs’ controllability and observability matrices 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.

The paper 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 desired training accuracy. This result in turn forms the basis to 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.

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 paper.

The computational framework of an RCN and its training procedure were 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], [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. In spite of 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 dressing its design were also reported. The performance of the ESP [14], [15], the memory capacity [16]–[18], and the control-theoretic approaches [24], [25]. For instance, the contracting maps, principles to realize RCNs with linear and nonlinear activations were studied in [26] leveraging

The main contributions of this paper 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 paper 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 \( v : M \to \mathbb{R} \) be an observation function, and in practice, we measure a discrete sequence of observations, say \( v[s(t_i)] \), 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 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 \((\phi, v)\) with \( \phi \in D(M) \), and \( v \in C^2(M, \mathbb{R}) \), it is a generic property that the map \( \Phi_{\phi,v,2m+1} : M \to \mathbb{R}^{2m+1} \), defined by \( \Phi_{\phi,v,2m+1}(s) = (v(s), v(\phi(s)), \ldots, v(\phi^{2m}(s))) \) is an embedding, where ‘generic’ means open and dense in \( C^1 \) topology.

From here on, we use \( \Phi_{2m+1} \) as an abbreviation for \( \Phi_{\phi,v,2m+1} \), and call \( 2m+1 \) as the ‘length’ of the time-delay embedding \( \Phi_{2m+1} \). Intuitively, Takens theorem states that for almost all pairs \((\phi, v)\) 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 Takens theorem holds, then by the definition of an embedding, the inverse function for the map \( \Phi_{2m+1} \) 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 \( \phi \) does, under a coordinate change of \( \Phi_{2m+1} \). The non-triviality of the construction of \( \psi_{2m+1} \) is that it forecasts a new observation when provided with the time-delayed observations \((v(s), v(\phi(s)), \ldots, v(\phi^{2m}(s)))\).

Namely, it holds that \( \psi_{2m+1}(v(s), v(\phi(s)), \ldots, v(\phi^{2m}(s))) = (v(s), v(\phi(s)), \ldots, v(\phi^{2m}(s))) \), which implies that if one learns the explicit representation of \( \psi_{2m+1} \), then the new observation, i.e., \( v(\phi^{2m+1}(s)) \), can be predicted based on the historical observations, \((v(s), \ldots, v(\phi^{2m}(s)))\). Specifically, if \( v = I_m \), then \( \psi_{2m+1} \) essentially predicts how the dynamical system defined by \( \phi \) is evolving on \( M \). For additional details on Takens theorem, see [28], [29].

#### B. The RCN dynamics and its training procedure

In this part, 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 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
\[ x[k + 1] = (1 - \alpha)x[k] + \alpha\sigma(Ax[k] + Bu[k]), \]
\[ y[k] = Cx[k] = \sum_{i=1}^{N} c_i x_i[k], \]
where \( x[k] \in \mathbb{R}^N \) denotes the state of the RCN and \( N \) 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; and \( A \in \mathbb{R}^{N \times N} \) and \( B \in \mathbb{R}^{N \times p} \) are pre-determined matrices denoting the connections in the RCN; \( C \in \mathbb{R}^{q \times N} \) is the weight matrix in output-layer to be trained, with \( c_i, i = 1, \ldots, N \) as its \( i \)th column; \( \alpha \in (0, 1) \) is called the ‘leakage rate’, and \( \sigma \) is an activation function that is applied to a vector component-wise.

A well-known result called the ‘leakage rate’, and \( \sigma \)

**Lemma 1.** Consider the dynamics of the RCN in (1). If \( \sigma \) is Lipschitz continuous with the Lipschitz constant \( L \), then the ESP holds if \( \|A\|_2 < \frac{1}{L} \), 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 \( w \)th 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 \( \tilde{y}[w; T] \) and the reference sequence \( \tilde{y}[w; T] \), that is,

\[ C = \arg\min_{C \in \mathbb{R}^{q \times N}} \sum_{k=w}^{T} \| C \left[ x[k] \right] - \tilde{y}[k] \|_2^2. \]

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 the following section, 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 1-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 one-dimensional time-series, i.e., \( u[k], y[k] \in \mathbb{R} \), and the framework is directly applicable to the multi-dimensional cases since the Takens theorem holds regardless of the dimension of the time-series.

1) 1-step ahead forecast: Suppose 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 1-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 \( i \)th component of the vector \( x[k] \), 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 \( \|A\|_2 < \frac{1}{L} \), the RCN acquires the ESP. This implies that there exists a \( \xi_0 \in \mathbb{N} \) such that, after the RCN evolves for \( \xi_0 \) steps, the effect of the initial condition on the solution trajectory becomes negligible. Therefore, for a fixed ‘washout’ length \( w > \xi_0 \), we can define \( \xi_{(i,w)} : \mathbb{R}^w \to \mathbb{R}^N \) such that

\[ x_i[w + j] = \varphi_i(x[j], u[j], \ldots, u[w + j - 1]) = \xi_{(i,w)}(u[j], u[j + 1], \ldots, u[w + j - 1]), \]

for all \( j = 0, \ldots, T - w \), where \( \xi_{(i,w)} \) can be treated as \( \varphi_i \) 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 \( \tilde{y}[w; T] \) and \( \hat{y}[w; T] \). Hence, training the RCN is equivalent to finding \( c_i \)'s such that

\[ u[w + j] \approx y[w + j] = \sum_{i=1}^{N} c_i x_i[w + j] \]

\[ = \sum_{i=1}^{N} c_i \varphi_i(x[j], u[j], u[j + 1], \ldots, u[w + j - 1]) \]

\[ = \sum_{i=1}^{N} c_i \xi_{(i,w)}(u[j], u[j + 1], \ldots, u[w + j - 1]), \]

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 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 \( \psi_w \) using nonlinear functions \( \xi_{(i,w)} \) for \( i = 1, \ldots, N \).

2) Multi-step ahead forecasting: Based on the idea of 1-step ahead forecast, we can extend the RCN to accomplish \( \tau \)-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 1-step ahead forecasting holds for \( \tau \)-step forecasting since the output of the RCN can be configured to approximate \( \psi^n_w = \psi_w \circ \cdots \circ \psi_w \) (\( \tau \) times) such that \( \psi^n_w(u[j; w + j - 1]) = u[j + \tau; w + \tau + \tau - 1] \). Hence, in this way, the RCN training can be viewed as approximating the last component of \( \psi^n_w \) by using \( \sum_{i=1}^{N} c_i \xi_{(i,w)} \).

**Remark 1.** The above explanation on multi-step 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, 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, 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 1-step/multi-steps 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 \( \sigma \) is Lipschitz continuous with the Lipschitz constant \( L \), the condition \( \|A\|_2 < \frac{1}{L} \) 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 \( \sigma \), we indeed have much freedom on the choice of \( \sigma \). In fact, the activation function can be as simple as a linear function.

The trade-off 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 \( \text{sigmoid} \), 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 \( u(0; T - 1) \subseteq \mathbb{R}^p \) and \( \tilde{y}(1; T) \subseteq \mathbb{R}^q \), 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 \( u[0; T - 1] \subseteq \mathbb{R}^p \) and \( \tilde{y}[1; T] \subseteq \mathbb{R}^q \), we say that the triplet, \( \hat{A} \in \mathbb{R}^{N \times N} \), \( \hat{B} \in \mathbb{R}^{N \times p} \), and \( \hat{C} \in \mathbb{R}^{q \times N} \), is an \( N \)-dimensional \( \epsilon \)-error realization of the pair \( (u[0; T - 1], \tilde{y}[1; T]) \) if the linear system,

\[
\begin{align*}
x[k + 1] &= \hat{A}x[k] + \hat{B}u[k], \\
y[k] &= \hat{C}x[k],
\end{align*}
\]

satisfies \( \sqrt{\sum_{k=1}^{T} ||y[k] - \tilde{y}[k]||^2} \leq \epsilon. \) 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 \( (\hat{A}, \hat{B}, \hat{C})_N \), 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 \( k \)-th Markov parameter of a realization \( (\hat{A}, \hat{B}, \hat{C})_N \) is a matrix of real numbers \( \gamma_k \in \mathbb{R}^{q \times p} \) defined by \( \gamma_k = \hat{C}A^kB \).

Definition 3 (Equivalent realizations). Two realizations \( (\hat{A}_1, \hat{B}_1, \hat{C}_1)_N \) and \( (\hat{A}_2, \hat{B}_2, \hat{C}_1)_{N_2} \) are said to be equivalent if \( \gamma_k^{(1)} = \gamma_k^{(2)} \) holds for all \( k = 0, 1, 2, \ldots \) where \( \gamma_k^{(1)} = C_1A_1^kB_1 \) and \( \gamma_k^{(2)} = C_2A_2^kB_2 \).

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

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

Next, we will describe the RCN training procedure through the use of a realization \( (\hat{A}, \hat{B}, \hat{C})_N \), 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., \( \sigma \) is the identity function, given by

\[
x[k + 1] = [(1 - \alpha)I + \alpha A]x[k] + \alpha Bu[k],
\]

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

\[
x[k + 1] = \hat{A}x[k] + \hat{B}u[k],
\]

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 $\alpha$, and the randomly generated matrices $\hat{A}$ and $\hat{B}$. Then, training the output layer of the RCN constructed using $\hat{A}$ and $\hat{B}$ is equivalent to finding $\hat{C}^*$ such that

$$\hat{C}^* = \arg\min_{\hat{C} \in \mathbb{R}^{N \times N}} E[(\hat{A}, \hat{B}, \hat{C})_N],$$

(6)

where $E[(\hat{A}, \hat{B}, \hat{C})_N] := \sqrt{\sum_{k=1}^{T_1} \| \hat{y}[k] - \hat{C} \hat{A}^k B u[k]\|_2^2}$ denotes the training error of the realization $(\hat{A}, \hat{B}, \hat{C})_N$.

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

$$\min_{\hat{C} \in \mathbb{R}^{N \times N_2}} E[(\hat{A}_2, \hat{B}_2, \hat{C})_{N_2}] \leq E[(\hat{A}_2, \hat{B}_2, \hat{C}_2)_{N_2}] = E[(\hat{A}_1, \hat{B}_1, \hat{C}_1)_{N_1}] < \epsilon,$$

which implies that if we train the RCN constructed using the $N_2$-dimensional matrices $\hat{A}_2 \in \mathbb{R}^{N_2 \times N_2}$ and $\hat{B}_2 \in \mathbb{R}^{N_2 \times p}$, the training error will be bounded above by $\epsilon$. 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 $\hat{A}$ in order for the RCN to be equipped with the ESP. To achieve this, we propose the notion of $\alpha$-stable realization.

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

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

$$\| \hat{A} \|_2 = \frac{1}{\alpha} (\| \hat{A} - (1 - \alpha) I \|_2 \leq \frac{1}{\alpha} (\| \hat{A} \|_2 + (1 - \alpha))$$

$$< \frac{1}{\alpha} (2\alpha - 1 + 1 - \alpha) = 1.$$  

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

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

**B. The irreducible stable realization**

Theoretically, if there exists one stable realization that achieves a 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 $(\hat{A}, \hat{B}, \hat{C})_N$ as modeled in (4), the controllability and the observability matrices are defined by

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

respectively.

Controllability and observability properties then lead to the characterization of an irreducible realization (see [33]).

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

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 $\alpha$-stable realization $(\hat{A}, \hat{B}, \hat{C})$, if $\text{rank}(W_N) = N < N$, then there exists an $\tilde{N}$-dimensional $\alpha$-stable realization $(\tilde{A}, \tilde{B}, \tilde{C})_{\tilde{N}}$ that is equivalent to $(\hat{A}, \hat{B}, \hat{C})_N$.

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

$$V^{-1}\hat{A}V = V^T\hat{A}V = \begin{bmatrix} V_1^T\hat{A}V_1 \\ V_2^T\hat{A}V_2 \end{bmatrix}.$$

Note that each column of $\hat{A}V_1$ 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 $V_2^T\hat{A}V_1 = 0$. Therefore, $V^{-1}\hat{A}V$ can be re-written as

$$V^{-1}\hat{A}V = \begin{bmatrix} V_1^T\hat{A}V_1 \\ 0 \end{bmatrix} := \begin{bmatrix} A_{11} \\ A_{12} \end{bmatrix}.$$

(7)

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

$$V^{-1}\hat{B} = V^T\hat{B} = \begin{bmatrix} V_1^T\hat{B} \\ 0 \end{bmatrix} := \begin{bmatrix} B_1 \\ 0 \end{bmatrix}.$$

(8)

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

$$\tilde{C}V(V^{-1}AV)^kV^{-1}B = \tilde{C}\hat{A}^k\hat{B}.$$
holds for all \( k = 0, 1, \ldots \). Furthermore, due to the structure provided by (7) and (8), it can be verified that

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

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

\[
\|A_{11}\|_2 = \sup_{x \in \mathbb{R}^N, \|x\|_2 = 1} x^\top A_{11}x = \sup_{x \in \mathbb{R}^N, \|x\|_2 = 1} x^\top V_1^\top \tilde{A}V_1x. \quad (9)
\]

Let \(y = V_1x\), then \(y\) is a vector in \(\mathbb{R}^N\) satisfying

\[
\|y\|_2^2 = y^\top y = (x_1v_1 + \cdots + x_Nv_N)(x_1v_1 + \cdots + x_Nv_N) = x_1^2 + \cdots + x_N^2 = \|x\|_2^2.
\]

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

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

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

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 \((\tilde{A}, \tilde{B}, \tilde{C})_{\tilde{N}}\) is irreducible, i.e., there exists no equivalent stable realization \((A, B, C)_N\) with \(N < \tilde{N}\), if and only if rank \(W_NG_N = N\).

**Proof.** We prove the theorem by proving the contraposition, i.e., \((A, B, C)_N\) is not irreducible if and only if rank \(W_NG_N < N\).

On the one hand, if \((\tilde{A}, \tilde{B}, \tilde{C})_{\tilde{N}}\) is not irreducible, then by Lemma 3 either rank \(W_N < N\) or rank \(G_N < N\). Therefore,

\[
\text{rank} (W_NG_N) \leq \min(\text{rank} (W_N), \text{rank} (G_N)) < N.
\]

On the other hand, if rank \(W_NG_N < N\), then, by Sylvester’s rank inequality, it holds that

\[
\text{rank} (W_N) + \text{rank} (G_N) - N \leq \text{rank} (W_NG_N) < N,
\]

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

As a consequence of Lemma 3, Lemma 4 and Theorem 2, the procedure for finding the irreducible stable realization that is equivalent to a given stable realization \((A, B, C)_N\) is described in Algorithm 1 where 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 \((\tilde{A}, \tilde{B}, \tilde{C})_{\tilde{N}}\)
while rank \((W_NG_N) < N\) do
  if rank \((W_N) < N\) then
    \(V_1 = \text{Orth} (W_N)\).
  else
    \(V_1 = \text{Orth} (G_N)\).
  end if
  \(\tilde{A} = V_1^\top \tilde{A}V_1, \tilde{B} = V_1^\top \tilde{B}, \tilde{C} = \tilde{C}V_1, \)
  \(N = \text{dim} \tilde{A}\).
  Compute \(W_N, G_N\) for \((\tilde{A}, \tilde{B}, \tilde{C})_N\).
end while
return \((\tilde{A}, \tilde{B}, \tilde{C})_{\tilde{N}}\).
end function

**Remark 4.** Each iteration in Algorithm 1 consists of computing the eigen-decomposition 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 to 20 iterations for the cases when \(N = 500, 1000\) 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 involves evaluating the performance of models with various hyper-parameters and choosing the model that yields the best result [27, 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 the previous section 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 Takens embedding, which provides a criterion to characterize the complexity of the underlying dynamics determined by \(u[0; T-1]\) and \(y[1; T]\).

From the theory of linear dynamical system (see [40]), it is a known fact that for any \(N\)-dimensional realization \((A, B, C)_N\) 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

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

\[
P^{-1} \tilde{B} = \begin{pmatrix}
\gamma_0 \\
\vdots \\
\gamma_{N-1}
\end{pmatrix}, \quad \tilde{C} P = (I_q, 0, \cdots, 0),
\]

(10)

where \(\theta_j, j = 0, 1, \ldots, N - 1\) are arbitrary constants and \(\gamma_j, j = 0, 1, \ldots, N - 1\) 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 \((A, B, C)_N\) is in the observable canonical form as in (10).

In this case, the dynamics of each component of \(x[k]\) is given by

\[
x[k + 1] = \begin{pmatrix}
x_2[k] + \gamma_0 u[k] \\
x_N[k] + \gamma_N u[k]
\end{pmatrix}, \quad \tilde{y}[k] = x_1[k],
\]

where \(x_i[k]\) is the \(i\)th component of \(x[k]\). Therefore, we have

\[
\tilde{y}[k + N] = x_1[k + N] = x_2[k + N - 1] + \gamma_0 u[k + N - 1] = \cdots = x_N[k + 1] + \sum_{i=0}^{N-1} \gamma_i u[k + N - 1 - i].
\]

(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 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 Takens theorem. On the other hand, if a linear realization of size \(N\) attains a training error of \(\epsilon\), then it implies that there exists a dynamical system evolving on a manifold of dimension \(\frac{1}{2}(N - 1)\) that approximately represents the underlying dynamics up to \(\epsilon\)-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 \(\psi_{2m+1}\) in 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 RCN dynamics with a nonlinear activation function is in general unavailable. In this case, the linear realization of the RCN offers a guideline towards 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 \((A, B, C)_N\), 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

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

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 desired training error performance.

A. The 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 Takens embedding.

We first introduce how the training data was generated and how the RCN was trained. For a fixed time-delay \(\tau\), 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 \(\tau\)-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 \(\tau\) from 1 to 50. For each \(\tau\), we randomly generated a \(N_0\)-dimensional 0.001-error realization with \(N_0 > \tau\) and computed its irreducible realization using Algorithm 1. Then, using the irreducible realization, we forecast the sequence \(\tau\)-step ahead for 2000 time steps. When training RCNs, we fixed training length as \(T = 1000\), and leakage rate as \(\alpha = 0.9\).

Figure 4a demonstrates the size of irreducible linear realization \(N_{\text{min}}\) with respect to \(\tau\) 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. Figure 4a shows that using Algorithm 1 we can trim a large RCN (dashed green line) into much smaller size (solid blue 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 (dashed red line) associated with the training dataset. Figure 4b provides the averaged mean squared errors (MSE) of the irreducible linear realization with respect to \(\tau\).
Fig. 1: Results of designing RCNs to forecast a time-delay system. (a) The size of irreducible 0.001-error realization versus length of time-delay. (b) The training and forecasting MSE of the irreducible 0.001-error realization versus length of time-delay.

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, given by

\[
\begin{align*}
\frac{dx}{dt} &= -y - z, \\
\frac{dy}{dt} &= x + ay, \\
\frac{dz}{dt} &= b + z(x - c),
\end{align*}
\]

where \(x, y, z\) are state variables and \(a, b, c\) 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. Figure 2a provides a demonstration of forecasting the \(x\)-component of Rössler system for 30-steps ahead.

We first randomly generate two RCNs with all hyper-parameters to be the same, but one with tanh activation function, the other one with linear activation function. The hyper-parameters 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 \(\|A\|_1\) with \(\lambda = 10^{-8}\).

Figures 2b presents the results of forecasting the time-series provided in Figure 2a with a mean-squared-error (MSE) of \(1.49 \times 10^{-3}\). Under the same setup as in Figure 2b, we varied the leakage rate \(\alpha\) and the activation function of the RCN, and present the corresponding results of MSE versus time-delay \((\tau)\) in Figures 2c and 2d respectively. From Figure 2c, we observed that \(\alpha = 0.05\) resulted in the best MSE across different cases of \(\tau\); and from Figure 2d, we observed that the RCN with linear activation function achieved similar performance as the RCNs with a nonlinear activation function when the time-delay \(\tau\) was small. Therefore, we used the same RCN as in Figure 2b, 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 Figure 2e with an MSE of \(2.97 \times 10^{-2}\).

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 \(2.54 \times 10^{-2}\). Figure 2f presents the result of using the irreducible RCN to forecast the same time-series generated by Rössler system for 30-steps ahead. Based on our empirical analysis in Section IV-C, the results in Figure 2f 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 part, 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 hyper-parameters 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 function. 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 10-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 hyper-parameters 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.
Fig. 2: (a) Experiment setup for using RCN to predict the time-series generated by a Rössler system for 30-steps ahead. (b) Forecasting the same time-series as (a), using tangent hyperbolic function. (c) The mean squared error (MSE) between the forecast output and target, versus different time-delays, using the same RCN as in (b) with different leakage rate. (d) The mean squared error (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 linear function. (f) Forecasting the same time-series as (a) using the reduced RCN computed from (e) through Algorithm II.

VI. CONCLUSIONS

In this paper, 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 hyper-parameters 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 $\alpha$-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


<table>
<thead>
<tr>
<th>N</th>
<th>Training MSE (mean ± std)</th>
<th>Forecasting MSE (mean ± std)</th>
<th>P(\text{linear} &gt; \max(\epsilon_{\text{tanh}}, \epsilon_{\text{sigmoid}}))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>linear (x10^-5)</td>
<td>tanh (x10^-5)</td>
<td>sigmoid (x10^-7)</td>
</tr>
<tr>
<td>50</td>
<td>0.11 ± 0.00</td>
<td>0.10 ± 0.00</td>
<td>1.55 ± 0.50</td>
</tr>
<tr>
<td>100</td>
<td>0.04 ± 0.00</td>
<td>0.04 ± 0.00</td>
<td>0.53 ± 0.13</td>
</tr>
<tr>
<td>150</td>
<td>3.87 ± 0.04</td>
<td>3.62 ± 0.04</td>
<td>0.32 ± 0.06</td>
</tr>
<tr>
<td>200</td>
<td>4.63 ± 0.00</td>
<td>4.52 ± 0.02</td>
<td>0.24 ± 0.04</td>
</tr>
<tr>
<td>250</td>
<td>0.89 ± 0.00</td>
<td>0.86 ± 0.00</td>
<td>0.20 ± 0.03</td>
</tr>
<tr>
<td>300</td>
<td>2.60 ± 0.00</td>
<td>2.51 ± 0.00</td>
<td>0.18 ± 0.02</td>
</tr>
<tr>
<td>350</td>
<td>0.02 ± 0.00</td>
<td>0.10 ± 0.00</td>
<td>0.11 ± 0.02</td>
</tr>
<tr>
<td>400</td>
<td>2.12 ± 0.00</td>
<td>2.11 ± 0.00</td>
<td>0.15 ± 0.01</td>
</tr>
<tr>
<td>450</td>
<td>4.60 ± 0.00</td>
<td>4.59 ± 0.00</td>
<td>0.14 ± 0.01</td>
</tr>
<tr>
<td>500</td>
<td>4.45 ± 0.00</td>
<td>4.36 ± 0.00</td>
<td>0.13 ± 0.01</td>
</tr>
</tbody>
</table>

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 hyper-parameters and the same randomly generated \( \hat{A} \) and \( \hat{B} \). The error of using RCN of different sizes to forecast the Rössler system in Section 14 for 10-steps ahead is provided in the table, where 2000 independent experiments are conducted for each N.


