

Reservoir Computing with Random and Optimized Time-Shifts

Enrico Del Frate,¹ Afroza Shirin,¹ and Francesco Sorrentino^{1, a)}

Mechanical Engineering Department, University of New Mexico, Albuquerque, NM 87131

We investigate the effects of application of random time-shifts to the readouts of a reservoir computer in terms of both accuracy (training error) and performance (testing error.) For different choices of the reservoir parameters and different ‘tasks’, we observe a substantial improvement in both accuracy and performance. We then develop a simple but effective technique to optimize the choice of the time-shifts, which we successfully test in numerical experiments.

We study how the accuracy and performance of a reservoir computer (RC) can be enhanced by application of different time-shifts to the RC readouts. Our numerical analysis shows an improvement for different parameters of the RC dynamics and for different ‘tasks’, such as reconstructing the attractors of several chaotic dynamical systems. For certain tasks, the attained improvement is of several orders of magnitude.

I. INTRODUCTION

A reservoir computer (RC) is a complex nonlinear dynamical system that is used for processing and analyzing empirical data, see e.g.^{1–11}, modeling of complex dynamical systems¹², speech recognition¹³, learning of context free and context sensitive languages^{14,15}, the reconstruction and prediction of chaotic attractors^{16–20}, image recognition²¹, control of robotic systems^{22–24}, predicting catastrophic critical transitions²⁵ and amplitude death in oscillating systems²⁶. A typical RC consists of a set of nodes coupled together to form a network. Each node of the RC evolves in time in response to an input signal that is fed into the reservoir. An output signal is then generated from the time evolutions of the RC nodes. In an RC, the output connections (those that connect the RC nodes to the output) are trained to produce a best fit between the output signal and a training signal related to the original input signal. On the other hand, the connections between the nodes of the reservoir are constant parameters of the system. As a result, RCs are easier to analyze than other machine learning tools for which all the connections are typically trained.

The performance of an RC depends on variety of factors such as nonlinearity of the nodal dynamics^{27,28}, network topology, sparsity of the connections and the presence of network symmetries²⁹, input signal and the dynamic range of the input signals³⁰ and time-delay structure of the RC^{5,8,11}. Experimental realizations of RCs have been proposed in^{6,9,11,31}, among other papers. Recent work has analyzed linear RCs^{32,33} and

pointed out a connection with the theory of dynamic mode decomposition³⁴.

A universal approximation theorem and its application to reservoir computers with stochastic inputs has been presented in³⁵. In^{33,36} it has been shown that a reservoir computer can perform as a universal representor of a dynamical system. Other papers have shown dramatic effects of tuning several parameters and hyper-parameters of RCs, see e.g.,^{29,37,38}. In particular, Ref.²⁹ investigated the effects of the sign of the weights associated with the network edges as well as the symmetries of the network topology and shown that network symmetries are usually undesirable in terms of the performance of RCs.

In this paper, we focus on the effects of time-shifts applied to the readouts of an RC. In the literature there has been documented improvements in RC performance by applying a single time-shift to all nodes³⁹, however, applying different time-shifts to individual nodes is new. References^{5,11} focused on the case that the time evolution of the RC obeys a delay differential equation. This is different from what we do here where the RC dynamics is described by an ordinary differential equation; once the RC dynamics is computed, different time-shifts are applied to the individual RC readouts.

In the first part of this paper, random time-shifts are applied at the readout of each node of an RC, which produces an improvement in both accuracy and performance. In the second part of this paper, a simple optimization technique is implemented to optimize the time-shift at each node in order to further improve the accuracy and performance of an RC. Optimizing the hyper-parameters of an RC is often done, but optimizing the time-shift at each node of an RC is more difficult due to the high-dimensional parameter space. Our numerical analysis shows that for different parameters of the RC dynamics, and for different ‘tasks’, an RC with time-shifts provides an increase in accuracy and performance.

II. METHODS

A. Reservoir Dynamics

We consider an RC modeled by the following nonlinear dynamical equations in continuous time⁴⁰,

$$\dot{\mathbf{r}}(t) = \gamma [-\mathbf{r}(t) + \tanh(\epsilon \mathbf{A} \mathbf{r}(t) + s(t) \mathbf{w})], \quad (1)$$

^{a)}Electronic mail: fsorrent@unm.edu

where $\mathbf{r}(t)$ is the N -dimensional state vector of the reservoir and $s(t)$ is the input signal, the N -dimensional symmetric adjacency matrix $A = \{A_{ij}\}$ describes the connectivity between the N nodes of the network and the N -dimensional vector \mathbf{w} are the weights by which the input signal is multiplied. In what follows we refer to the time evolutions $\mathbf{r}(t)$ as the readouts of the RC. In this paper we set $N = 100$. The adjacency matrix A is constructed such that it is symmetric and its off-diagonal entries are uniformly drawn at random from the interval $[0, 1]$. The entries on the main diagonal of the matrix A are all set to be equal to $\beta < 0$, where the scalar β is negative enough to ensure that all the eigenvalues of the matrix A are less than 0. The variable parameter ϵ is used to tune the spectral radius of the matrix A . The entries of the vector \mathbf{w} are all chosen to be 1. The variable parameter $\gamma > 0$ determines the time-scale on which the RC dynamics evolves.

The underlying process we want to model may evolve in time based on a set of deterministic (chaotic) equations, such as the equations of the Lorenz chaotic system, in the variables $x(t), y(t), z(t)$ (See Eq. (16).) One task that can be given to the RC is to reconstruct the time evolution of the training signal, e.g. $y(t)$ from knowledge of the input signal, e.g., $x(t)$. In this paper we will consider several similar tasks for which the time series are generated by various chaotic systems. An example of chaotic higher-dimensional system we will use in this paper is the Lorenz96 system⁴¹. For this system, we see particularly strong benefits of introducing the time-shifts.

B. Training and Testing Error of The Reservoir Computer

In order to examine the accuracy of the RC relative to the dynamical system it is modeling, we need to quantify how well the reservoir is able to reproduce the training signal $g(t)$ from knowledge of the input signals $s(t)$. An RC driven by the input signal has three phases: the transient phase which is from $t_0 = 0$ to t_1 , the training phase which is from t_1 to t_2 , and the testing phase which is from t_2 to t_3 .

During the training phase $[t_1 \ t_2]$ the readouts from each node are recorded, discretized, and combined in a $T \times (N + 1)$ matrix,

$$\Omega = \begin{bmatrix} r_1(1) & r_2(1) & \dots & r_N(1) & 1 \\ r_1(2) & r_2(2) & \dots & r_N(2) & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ r_1(T) & r_2(T) & \dots & r_N(T) & 1 \end{bmatrix} \quad (2)$$

Here, N is the number of nodes in the RC and T is the number of time-steps recorded in the interval $[t_1 \ t_2]$. We add a column whose entries are all ones to account for any constant offset. The fit $\mathbf{h} = [h(1), h(2), \dots, h(T)]$

to the training signal $\mathbf{g} = [g(1), g(2), \dots, g(T)]$ is equal to,

$$h(t) = \sum_{i=1}^N \kappa_i r_i(t) + \kappa_{N+1} \quad (3)$$

(or, equivalently, in vectorial form $\mathbf{h} = \Omega \boldsymbol{\kappa}$), where the vector $\boldsymbol{\kappa} = [\kappa_1, \kappa_2, \dots, \kappa_{N+1}]$, which contains a set of unknown coefficients to be determined. The weight vector $\boldsymbol{\kappa}$ is obtained by minimizing the linear least square fit problem,

$$\min_{\boldsymbol{\kappa}} \|\Omega \boldsymbol{\kappa} - \mathbf{g}\|_2^2. \quad (4)$$

The analytic solution of the problem is given by,

$$\boldsymbol{\kappa} = (\Omega^T \Omega)^{-1} \Omega^T \mathbf{g}, \quad (5)$$

When the matrix Ω is super-collinear (columns are highly linearly dependent or $T \gg N$), the inverse of the matrix $\Omega^T \Omega$ is difficult to compute numerically. To avoid this, we estimate the weight vector $\boldsymbol{\kappa}$ as a solution of the linear least square fit problem with ridge regression,

$$\min_{\boldsymbol{\kappa}} \|\Omega \boldsymbol{\kappa} - \mathbf{g}\|_2^2 + \eta \|\boldsymbol{\kappa}\|_2^2, \quad (6)$$

where $\eta > 0$ is a small positive number. The solution of the above problem can be computed as,

$$\boldsymbol{\kappa}_{\text{ridge}} = (\Omega^T \Omega + \eta I)^{-1} \Omega^T \mathbf{g}. \quad (7)$$

Here I is the identity matrix of size N . From this, we can compute the training error,

$$\Delta_{tr} = \frac{\langle \Omega \boldsymbol{\kappa}_{\text{ridge}} - \mathbf{g} \rangle}{\langle \mathbf{g} \rangle}, \quad (8)$$

where the notation $\langle \mathbf{X} \rangle = \sqrt{\frac{1}{T} \sum_{i=1}^T (X(i) - \mu)^2}$ for \mathbf{X} any T -dimensional vector and $\mu = \frac{1}{T} \sum_{i=1}^T X(i)$.

A fundamental measure of the performance of an RC is the testing error. The testing error is defined as,

$$\Delta_{ts} = \frac{\langle \tilde{\Omega} \boldsymbol{\kappa}_{\text{ridge}} - \tilde{\mathbf{g}} \rangle}{\langle \tilde{\mathbf{g}} \rangle}, \quad (9)$$

where $\tilde{\mathbf{g}}(t)$ is the testing signal we want to estimate, $\tilde{\Omega}$ contains the time evolutions from the RC over the time interval $[t_2 \ t_3]$, and $\boldsymbol{\kappa}_{\text{ridge}}$ is the same coefficient vector we found in the training phase.

C. Initial setting of the reservoir parameters γ and ϵ

In Secs. III and IV we will consider the effects of application of time shifts to a well performing Reservoir Computer, meaning that the RC has been preliminarily optimized based on current state-of-the-art practices. However, we stress out that we have seen similar improvements when the RC is not optimized.

Our preliminary optimization consists of two steps: (i) optimization in the coefficient γ and (ii) optimization in the coefficient ϵ , see Eq. (1). We first discuss (i) and then (ii). For each one of the tasks, in order to pick a best value of γ , we set $\epsilon = 1$ and compute the training error Δ_{tr} as a function of γ in the interval $[0, 5]$; we then pick the value of γ that minimizes Δ_{tr} . We keep γ at the selected value and investigate the effect of varying ϵ . In order to pick the best value of ϵ we compute the memory capacity defined as follows⁴²,

$$MC = \sum_{\tau=1}^{\infty} MC_{\tau}, \quad (10)$$

where

$$MC_{\tau} = \frac{\text{cov}^2(x(t-\tau), h(t))}{\text{var}(x(t))\text{var}(h(t))}, \quad \tau \in \mathbb{N} \quad (11)$$

and select the value of ϵ that maximizes the MC.

This procedure (optimization in γ followed by optimization in ϵ) is illustrated in Fig. 1 for the case of the Lorenz96 system⁴¹, see also Eq. (15) below. From (A) we see that the value of γ that minimizes the training error is approximately equal to 0.9. We then fix $\gamma = 0.9$ and vary ϵ , which is shown in (C). For completeness, the testing error is shown in (B). We see that the value of ϵ that maximizes the memory capacity is approximately equal to 0.8. Analogous procedures are implemented for the cases of the Lorenz system and of the Hindmarsh-Rose system, which we discuss later in Sec. III. For the Lorenz system we obtain $\gamma = 1.65$ and $\epsilon = 1$. For the Hindmarsh-Rose system we obtain $\gamma = 0.9$ and $\epsilon = 0.8$.

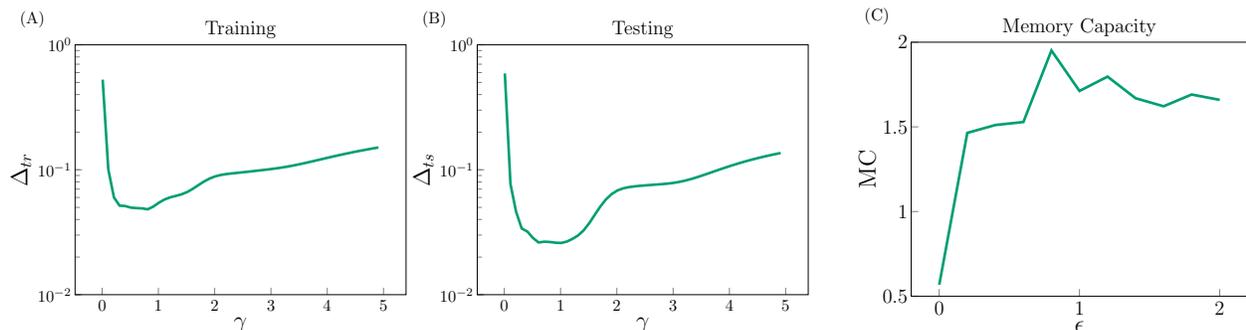


FIG. 1. Lorenz96 attractor. (A) Plot of the training error Δ_{tr} vs γ . The value of γ that minimizes the training error is approximately equal to 0.9. (B) Testing error vs γ , with a trend similar to that seen in (A). (C) Memory Capacity vs ϵ with γ set equal to 0.9. We find that the value of ϵ that maximizes the memory capacity is approximately equal to 0.8.

III. APPLICATION OF RANDOM TIME-SHIFTS

In this section we describe application of time-shifts to the individual readouts of a RC. We will see that introduction of these time-shifts is beneficial even when these are randomly chosen. Optimized time-shifts are considered in Sec. IV. Here our choice of random time-shifts is consistent with the choice of a random topology for the connectivity of the RC network, which is commonly assumed in the literature (see also our construction of the adjacency matrix A in section II.) This is typically done to show that RCs can be effective independent of the details of their implementation. We remove the assumption of randomly chosen time-shifts in Sec. IV.

For each individual task, we compute the timescale $\bar{\tau}$ of each individual oscillator system, defined as the time at which the system autocorrelation function decays to one half of its value at time zero. For the Lorenz system we find $\bar{\tau} = 0.3$; for the Hindmarsh-Rose system we find $\bar{\tau} = 0.46$; and for the Lorenz96 system we find $\bar{\tau} = 0.19$.

Subsequently, for each task, the individual time shifts τ_i are taken to be uniformly distributed random numbers in the interval $[0, \alpha\bar{\tau}]$, where α is a tunable parameter.

Finally, the reservoir readout at node i is shifted $r_i \mapsto r_i(t - \tau_i)$. The motivation for application of the time shifts is the observation that under general conditions the RC readouts $r_1(t), r_2(t), \dots$ appear to be ‘synchronized’¹⁴³, which significantly reduces the ability of fitting the training signal. By introducing time shifts, this synchronization can be broken.

The fit signal $\mathbf{h}(t)$ is written as a linear combination of the individual readouts,

$$\mathbf{h}(t) = \Omega_{\text{delay}} \boldsymbol{\kappa}_{\text{ridge}} \quad (12)$$

where $\boldsymbol{\kappa}_{\text{ridge}}$ in this case is computed as,

$$\boldsymbol{\kappa}_{\text{ridge}} = (\Omega_{\text{delay}}^T \Omega_{\text{delay}} + \eta I)^{-1} \Omega_{\text{delay}}^T \mathbf{g} \quad (13)$$

and Ω_{delay} is computed as,

$$\Omega_{\text{delay}} = \begin{bmatrix} r_1(1 - \tau_1) & r_2(1 - \tau_2) & \dots & r_N(1 - \tau_N) & 1 \\ r_1(2 - \tau_1) & r_2(2 - \tau_2) & \dots & r_N(2 - \tau_N) & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ r_1(T - \tau_1) & r_2(T - \tau_2) & \dots & r_N(T - \tau_N) & 1 \end{bmatrix} \quad (14)$$

In the testing phase the same times-shifts used in the training phase, are applied to compute $\tilde{\Omega}_{\text{delay}}$. Then the training and testing errors are computed by using Eq. (8) and (9), respectively.

In the rest of this section, we provide evidence of the strong benefits of applying randomly chosen time shifts, rather than presenting a principle way of selecting them. In Section IV we present an optimization approach that can be used to guide the selection of the time-shifts.

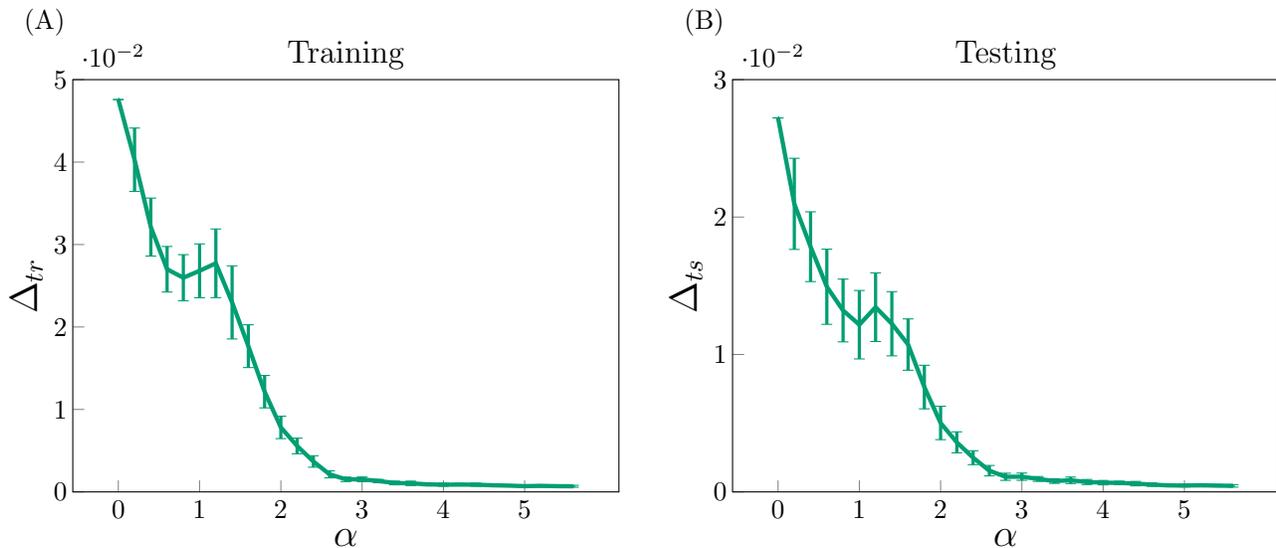


FIG. 2. Lorenz96 system. The training error (A) and testing error (B) vs α . The parameter α controls the interval over which the random time-shifts are taken $[0, \bar{\tau}\alpha]$, where $\bar{\tau}$ is the characteristic time-scale of the Lorenz task and was found to be 0.19. Error bars indicate the standard deviation over 50 iterations where each iteration corresponds to a different selection of the random time-shifts τ_i , $\gamma = 0.9$, $\epsilon = 0.8$.

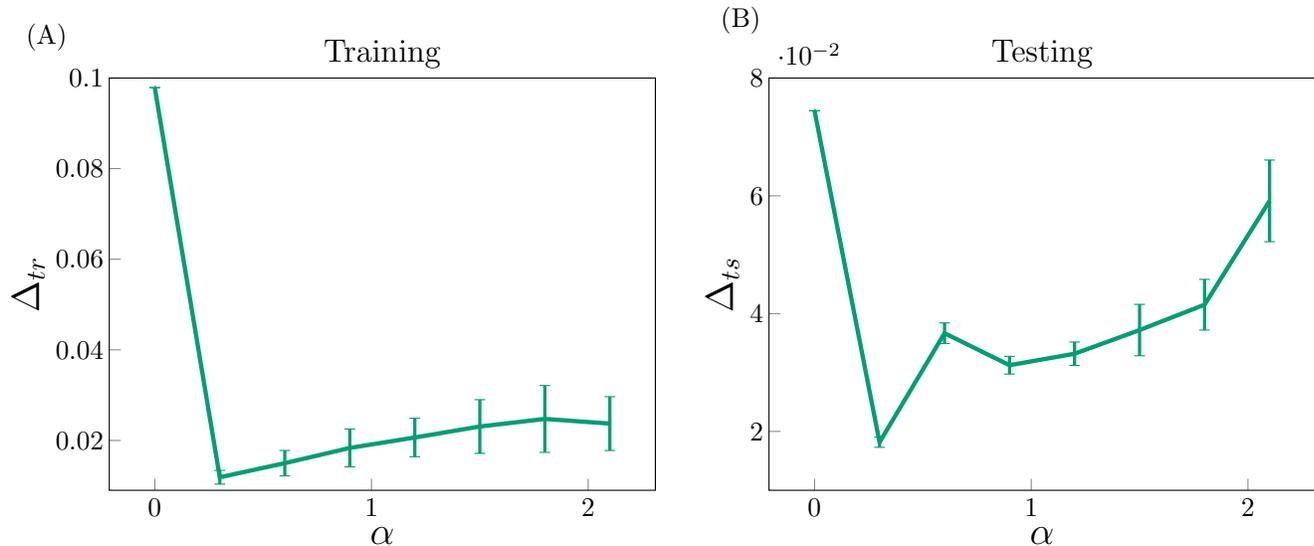


FIG. 3. Lorenz attractor. The training error (A) and testing error (B) vs α . The parameter α controls the interval over which the random time-shifts are taken $[0, \bar{\tau}\alpha]$, where $\bar{\tau}$ is the characteristic time-scale of the Lorenz task and was found to be 0.3. Error bars indicate the standard deviation over 50 iterations where each iteration corresponds to a different selection of the random time-shifts τ_i , $\gamma = 1.3$, $\epsilon = 2$.

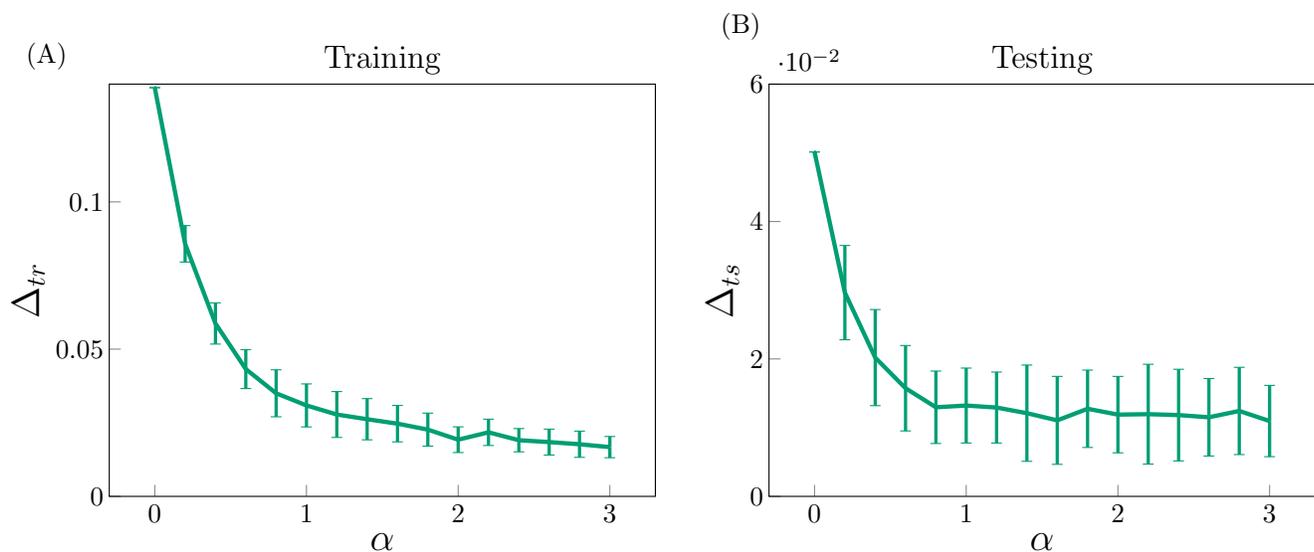


FIG. 4. Hindmarsh-Rose attractor. The training error (A) and testing error (B) vs α . The parameter α controls the interval over which the random time-shifts are taken $[0, \bar{\tau}\alpha]$, where $\bar{\tau}$ is the characteristic time-scale of the Lorenz task and was found to be 0.46. Error bars indicate the standard deviation over 50 iterations where each iteration corresponds to a different selection of the random time-shifts τ_i , $\gamma = 1.65$, $\epsilon = 1$.

The advantage of introducing random time-shifts is discussed in what follows for the case of three different ‘tasks’, which we simply refer to as the chaotic Lorenz 96 system, the Lorenz system, and the Hindmarsh-Rose

system.

A. Lorenz96 System Task

The Lorenz96 chaotic system is modeled by the following set of equations,

$$\dot{x}_i(t) = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F \quad (15)$$

where $i = 1, 2, \dots, M$ for which we assume: $x_{-1} = x_{M-1}, x_0 = x_M, x_{M+1} = x_1, F = 8, M = 4$.

We take the input signal $s(t) = x_1(t)$ and the training signal $g(t) = x_4(t)$. In the simulations the transient phase takes place from 0 to $t_1 = 1000$, the training phase is from $t_1 = 1000$ to $t_2 = 1100$, and the testing phase is from $t_2 = 1100$ to $t_3 = 1200$.

In Fig. 2, we plot the training (Δ_{tr}) and testing error (Δ_{ts}) vs α . The figure shows a substantial improvement in both Δ_{tr} and Δ_{ts} when α is increased from 0 to 5. In particular, we see that the mere application of random time shifts to the readouts of a nonlinear reservoir computer leads to a reduction of both the training and testing error of roughly two orders of magnitude.

B. Lorenz System Task

The Lorenz chaotic system is modeled by the following set of equations,

$$\begin{aligned} \dot{x}(t) &= c_1(y(t) - x(t)) \\ \dot{y}(t) &= x(t)(c_2 - z(t)) - y(t) \\ \dot{z}(t) &= x(t)y(t) - c_3z(t) \end{aligned} \quad (16)$$

with $c_1 = 10, c_2 = 28$ and $c_3 = 8/3$.

For this task, the $x(t)$ component is used as the input signal $s(t)$. The $y(t)$ component is used as the training signal. In simulation, the transient phase is set from 0 to $t_1 = 600$, the training phase from $t_1 = 600$ to $t_2 = 610$ and the testing phase from $t_2 = 610$ to $t_3 = 615$.

In Fig. 3, we plot the training (Δ_{tr}) and testing error (Δ_{ts}) vs α . For all values of $\alpha > 0$ we see a reduction in both the training and testing errors compared to the case that $\alpha = 0$, which corresponds to the case in which time-shifts are not applied. In particular, we see that both the lowest training error and the lowest testing error are achieved at intermediate values of α , with the best reduction for α approximately equal to 0.2. For larger values of α we see an increase of both the training and the testing error.

C. Hindmarsh-Rose System Task

The Hindmarsh-Rose chaotic system is modeled by the following equations,

$$\begin{aligned} \dot{x}(t) &= y(t) + \phi[x(t)] - z(t) + 1 \\ \dot{y}(t) &= \psi[x(t)] - y(t) \\ \dot{z}(t) &= 5 \times 10^{-3}(4(x(t) + 8/5) - z(t)) \end{aligned} \quad (17)$$

where,

$$\begin{aligned} \phi[x(t)] &= -x^3 + 3x^2 \\ \psi[x(t)] &= 1 - 5x^2. \end{aligned}$$

For this task, the $x(t)$ component is used as the input signal $s(t)$. The $y(t)$ component is used as the training signal. In these simulations the transient phase takes place from 0 to $t_1 = 1000$, the training phase from $t_1 = 1000$ to $t_2 = 1010$, and the testing phase from $t_2 = 1010$ to $t_3 = 1015$.

In Fig. 4, we plot the training (Δ_{tr}) and testing error (Δ_{ts}) vs α . For this case we see a substantial reduction in both the training error and testing error as we increase α . The improvement seen in both Δ_{tr} and Δ_{ts} when α is increased from 0 to 3 is roughly of one order of magnitude.

IV. OPTIMIZATION OF TIME-SHIFTS

In this section, we describe a method to select the time-shifts that minimizes the training error. This method requires calculation of the time derivative of the reservoir response. We proceed under the assumption that all the time shifts are small. This assumption may be confirmed or not after computation of the optimized time-shifts. However, we decide to retain this assumption for the following two reasons: (i) it allows a simple solution to the optimization problem and (ii) even if the assumption is not verified by the optimized solution, we still hope that it will improve the RC performance with respect to either the case of no time-shifts or random time-shifts. In what follows, we will test (ii) numerically for different choices of tasks and RC parameters. We will see that often times our strategy to optimize the time shifts overperforms random time shifts.

After applying small time-shifts to the individual readouts of the RC, a first order Taylor expansion yields,

$$r_i(t - \tau_i) \approx r_i(t) - \tau_i \dot{r}_i(t). \quad (18)$$

Now the fit signal $\mathbf{h}(t)$ in Eq. (12) can be written as,

$$\mathbf{h}(t) = \sum_{i=1}^N \kappa_i r_i(t) + \kappa_{N+1} + \sum_i \lambda_i \dot{r}_i(t), \quad (19)$$

where $\lambda_i = -\kappa_i \tau_i$. In other words,

$$\mathbf{h} = [\Omega_r \quad \Omega_{\dot{r}}] \begin{bmatrix} \boldsymbol{\kappa} \\ \boldsymbol{\lambda} \end{bmatrix} = \Omega_{\text{opt}} \begin{bmatrix} \boldsymbol{\kappa} \\ \boldsymbol{\lambda} \end{bmatrix}, \quad (20)$$

where $\Omega_r \equiv \Omega, \boldsymbol{\lambda} = [\lambda_1, \lambda_2, \dots, \lambda_N]$ and

$$\Omega_{\dot{r}} = \begin{bmatrix} \dot{r}_1(1) & \dot{r}_2(1) & \dots & \dot{r}_N(1) \\ \dot{r}_1(2) & \dot{r}_2(2) & \dots & \dot{r}_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{r}_1(T) & \dot{r}_2(T) & \dots & \dot{r}_N(T) \end{bmatrix}. \quad (21)$$

The optimal coefficient vector can be found by solving the linear square fit with ridge regression,

$$\begin{bmatrix} \boldsymbol{\kappa}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = (\Omega_{\text{opt}}^T \Omega_{\text{opt}} + \eta I)^{-1} \Omega_{\text{opt}}^T \mathbf{g} \quad (22)$$

with optimized time-shifts given by,

$$\tau_i^* = -\lambda_i^* / \kappa_i^*, \text{ for } i = 1, 2, \dots, N. \quad (23)$$

The fit signal with respect to the optimized time-shifts is,

$$\mathbf{h}^* = \Omega_{\text{shift}}^* \boldsymbol{\kappa}^*, \quad (24)$$

where Ω_{shift}^* is obtained by replacing τ_i by τ_i^* in Eq. (14) and the training error is computed by Eq. (8). The optimized time-shifts $\boldsymbol{\tau}^*$ and $\boldsymbol{\kappa}^*$ are then used to compute $\tilde{\Omega}_{\text{shift}}^*$ and the testing error. In all of our simulation in this article we set the ridge regression parameter η to 10^{-6} .

As we will see, though the optimization method is based on a first order approximation, it presents the main advantages that it is simple to compute numerically and it improves the accuracy and performance of the RC, compared to the case that no shifts are applied.

A. Lorenz96 System Task

Hereafter, we obtain the optimized time-shifts by the optimization method discussed in this section. The optimized time-shifts are applied to the individual nodes of the RC and the training and testing errors are computed.

Our best results are obtained for the case of the Lorenz96 System. A comparison between application of random time-shifts and optimized time-shifts is presented in Fig. 5. In this case, optimized time-shifts perform much better than random time-shifts. In particular, for low values of γ we see an improvement of many orders of magnitude, and a strong advantage of optimized time shifts is seen for all value of γ . Note also that random time-shifts still present a substantial improvement with respect to the case in which time-shifts are not used (compare Fig. 2 and Fig. 5.)

B. Lorenz System Task

In Fig. 6, we plot the training error (Δ_{tr}) and the testing error (Δ_{ts}) vs γ for different RC configurations. For $\gamma > 3$ we see that the training error for the case of optimized time-shifts is lower by roughly one order of magnitude. However, in terms of testing error we do not see a benefit of using optimized time-shifts. We wish to emphasize that both random and optimized time-shifts present a substantial improvement with respect to the case in which time-shifts are not used (compare Fig. 3 and Fig. 6.)

C. Hindmarsh-Rose System Task

A comparison between application of random time-shifts and optimized time-shifts is presented in Fig. 7 for the case of the Hindmarsh-Rose system. Fig. 7 (A) is a plot of the training error (Δ_{tr}) vs γ , showing that for most values of γ the RC accuracy is improved with optimized time-shifts. Fig. 7 (B) is a plot of the testing error (Δ_{ts}) vs γ , showing that for most values of γ in the range $1 < \gamma < 3$ the RC performance is improved with optimized time-shifts.

V. CONCLUSION

This paper discussed the benefits associated with application of time shifts to the readouts of a reservoir computer. In all of our numerical experiments, we preliminarily optimize the RC parameters so to ensure we are working with a well performing reservoir. However, our results hold for generic RCs.

For different ‘tasks’, we observe that application of randomly chosen time shifts to the reservoir readouts leads to a substantial improvement in both accuracy (training error) and performance (testing error) compared to the case in which time shifts are not used. The choice of random time shifts is consistent with the choice of a random topology for the connectivity of the RC network, which is commonly assumed in the literature. We see that the improvement observed is achieved independent of the particular selection of the time shifts. A further reduction in accuracy and performance is obtained when the time-shifts are computed by using a simple optimization approach. A case for which application of random and optimized time-shifts was particularly beneficial is that of the Lorenz96 system (see Figs. 2 and 5.)

The method we use to optimize the time-shifts is very simple and at the same time, effective. Optimization methods such as Particle Swarm⁴⁴, Simulated Annealing⁴⁵, etc could be used to compute better approximations to the optimal time-shifts but these other optimization methods typically require much higher computational complexity due to the large parameter space (in our case, a total of 100 time-shifts). On the other hand, the method we presented in this paper is easily scalable.

Our work may point out to a deeper connection with Taken’s Embedding Theorem⁴⁶, which states that a chaotic attractor can be reconstructed from a single ‘readout’ function of the underlying dynamical system and linearly spaced delayed observations of this only readout function. Here we are using N readouts and applying a different delay to each one of them. Exploring in more detail applications of Taken’s Embedding Theorem to reservoir computers provides a promising direction for future investigation.

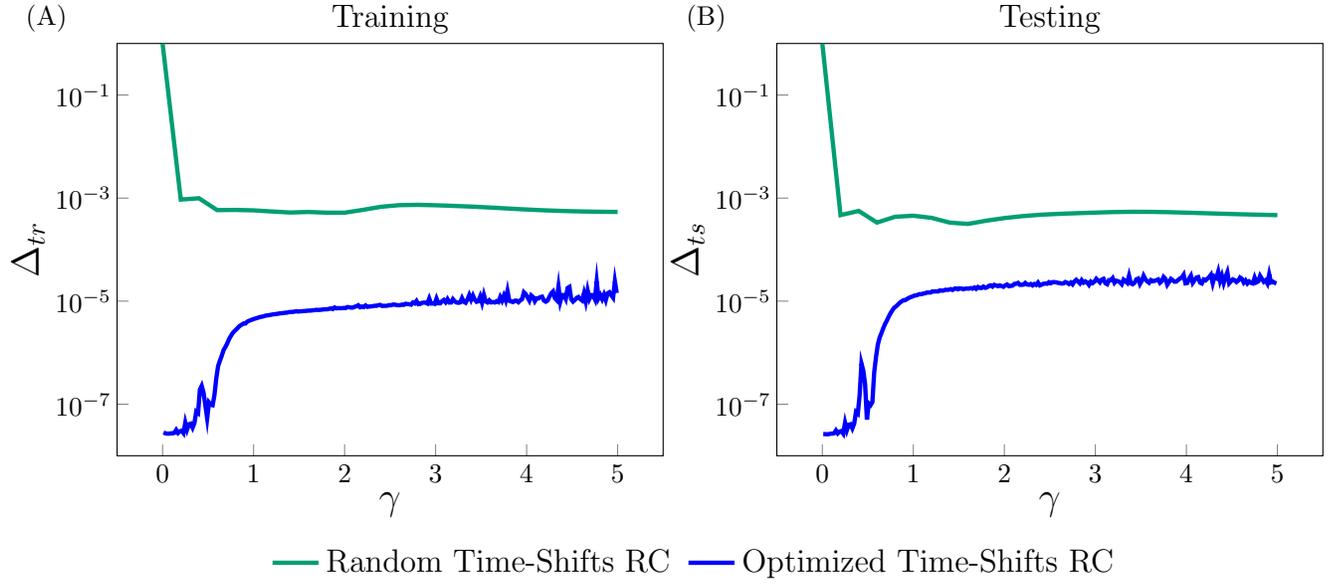


FIG. 5. Lorenz96 system. The training error (A) and testing error (B) vs γ for both the cases of: randomly drawn time shifts and optimized time shifts. Here $\epsilon = 0.8$ and $\alpha = 4$.

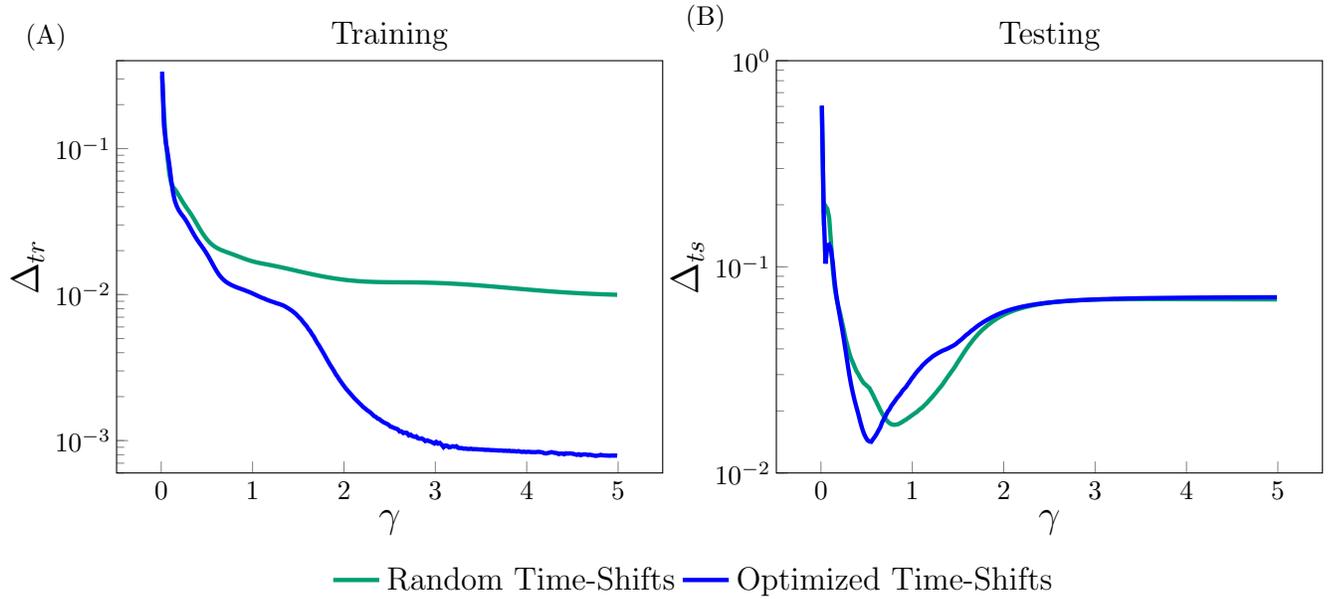


FIG. 6. Lorenz attractor. The training error (A) and testing error (B) vs γ are shown for the cases of randomly drawn timeshifts and optimized time-shifts. Here $\epsilon = 2$ and $\alpha = 0.25$.

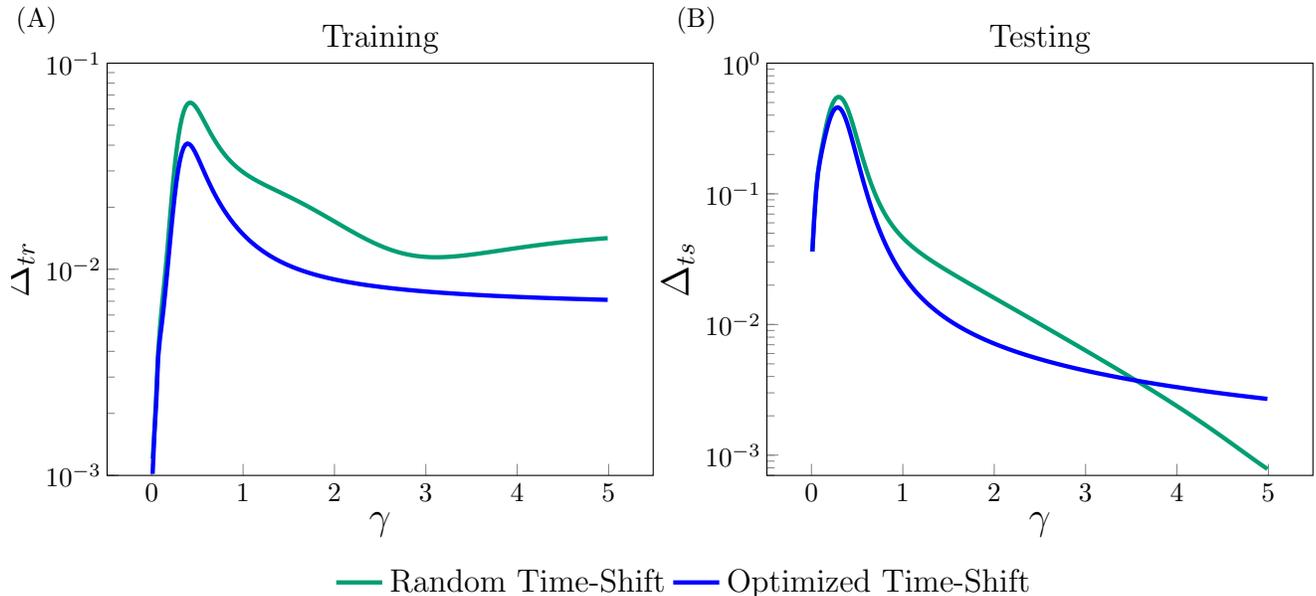


FIG. 7. Hindmarsh-Rose attractor. The training error (A) and testing error (B) vs γ for both the cases of randomly drawn time-shifts and optimized time-shifts. Here $\epsilon = 1$ and $\alpha = 2.5$.

ACKNOWLEDGEMENT

The authors thank Lou Pecora and Tom Carroll for insightful conversations on the subject of Reservoir Computers. This research was supported by NIH (NIBIB) grant 1R21EB028489-01A1.

AUTHOR DECLARATIONS

The authors have no conflicts to disclose.

DATA AVAILABILITY

The data that supports the findings of this study are available within the article.

REFERENCES

- ¹Herbert Jaeger. The “echo state” approach to analysing and training recurrent neural networks—with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.
- ²Benjamin Schrauwen, David Verstraeten, and Jan Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th european symposium on artificial neural networks*. p. 471-482 2007, pages 471–482, 2007.
- ³Thomas Natschläger, Wolfgang Maass, and Henry Markram. The “liquid computer”: A novel strategy for real-time comput-

ing on time series. *Special issue on Foundations of Information Processing of TELEMATIK*, 8(ARTICLE):39–43, 2002.

- ⁴Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.
- ⁵Romain Martinenghi, Sergei Rybalko, Maxime Jacquot, Yanne K Chembo, and Laurent Larger. Photonic nonlinear transient computing with multiple-delay wavelength dynamics. *Physical review letters*, 108(24):244101, 2012.
- ⁶Daniel Brunner, Miguel C Soriano, Claudio R Mirasso, and Ingo Fischer. Parallel photonic information processing at gigabyte per second data rates using transient states. *Nature communications*, 4:1364, 2013.
- ⁷Kohei Nakajima, Helmut Hauser, Tao Li, and Rolf Pfeifer. Information processing via physical soft body. *Scientific reports*, 5:10487, 2015.
- ⁸Michiel Hermans, Miguel C Soriano, Joni Dambre, Peter Bienstman, and Ingo Fischer. Photonic delay systems as machine learning implementations. *Journal of Machine Learning Research*, 2015.
- ⁹Quentin Vinckier, François Duport, Anteo Smerieri, Kristof Vandoorne, Peter Bienstman, Marc Haelterman, and Serge Massar. High-performance photonic reservoir computer based on a coherently driven passive cavity. *Optica*, 2(5):438–446, 2015.
- ¹⁰François Duport, Anteo Smerieri, Akram Akrouf, Marc Haelterman, and Serge Massar. Fully analogue photonic reservoir computer. *Scientific reports*, 6:22381, 2016.
- ¹¹Laurent Larger, Antonio Baylón-Fuentes, Romain Martinenghi, Vladimir S Udaltsov, Yanne K Chembo, and Maxime Jacquot. High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification. *Physical Review X*, 7(1):011015, 2017.
- ¹²Johan AK Suykens, Joos PL Vandewalle, and Bart L de Moor. *Artificial neural networks for modelling and control of non-linear systems*. Springer Science & Business Media, 2012.
- ¹³James P Crutchfield, William L Ditto, and Sudeshna Sinha. Introduction to focus issue: intrinsic and designed computation: information processing in dynamical systems—beyond the digi-

- tal hegemony, 2010.
- ¹⁴Paul Rodriguez. Simple recurrent networks learn context-free and context-sensitive languages by counting. *Neural computation*, 13(9):2093–2118, 2001.
- ¹⁵Felix A Gers and E Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.
- ¹⁶Zhixin Lu, Brian R Hunt, and Edward Ott. Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(6):061104, 2018.
- ¹⁷Piotr Antonik, Marvyn Gulina, Jaël Pauwels, and Serge Massar. Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronization and cryptography. *Physical Review E*, 98(1):012215, 2018.
- ¹⁸Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80, 2004.
- ¹⁹Jaideep Pathak, Zhixin Lu, Brian R Hunt, Michelle Girvan, and Edward Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.
- ²⁰Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical review letters*, 120(2):024102, 2018.
- ²¹Azarakhsh Jalalvand, Kris Demuyne, Wesley De Neve, and Jean-Pierre Martens. On the application of reservoir computing networks for noisy image recognition. *Neurocomputing*, 277:237–248, 2018.
- ²²Alex Graves, Douglas Eck, Nicole Beringer, and Juergen Schmidhuber. Biologically plausible speech recognition with lstm neural nets. In *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, pages 127–136. Springer, 2004.
- ²³Tony Robinson. An application of recurrent nets to phone probability estimation. *IEEE transactions on Neural Networks*, 5(2), 1994.
- ²⁴Mantas Lukoševičius, Herbert Jaeger, and Benjamin Schrauwen. Reservoir computing trends. *KI-Künstliche Intelligenz*, 26(4):365–371, 2012.
- ²⁵Ling-Wei Kong, Hua-Wei Fan, Celso Grebogi, and Ying-Cheng Lai. Machine learning prediction of critical transition and system collapse. *Physical Review Research*, 3(1):013090, 2021.
- ²⁶Rui Xiao, Ling-Wei Kong, Zhong-Kui Sun, and Ying-Cheng Lai. Predicting amplitude death with machine learning. *Physical Review E*, 104(1):014205, 2021.
- ²⁷Joni Dambre, David Verstraeten, Benjamin Schrauwen, and Serge Massar. Information processing capacity of dynamical systems. *Scientific reports*, 2:514, 2012.
- ²⁸Afroza Shirin, Isaac S Klickstein, and Francesco Sorrentino. Stability analysis of reservoir computers dynamics via lyapunov functions. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(10):103147, 2019.
- ²⁹Thomas L Carroll and Louis M Pecora. Network structure effects in reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(8):083130, 2019.
- ³⁰David Verstraeten and Benjamin Schrauwen. On the quantification of dynamics in reservoir computing. In *International Conference on Artificial Neural Networks*, pages 985–994. Springer, 2009.
- ³¹Forrest Sheldon and Francesco Caravelli. The computational capacity of mem-lrc reservoirs. In *Proceedings of the Neuro-inspired Computational Elements Workshop*, pages 1–4, 2020.
- ³²Stephen Boyd and Leon Chua. Fading memory and the problem of approximating nonlinear operators with volterra series. *IEEE Transactions on circuits and systems*, 32(11):1150–1161, 1985.
- ³³Erik 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: An Interdisciplinary Journal of Nonlinear Science*, 31(1):013108, 2021.
- ³⁴Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.
- ³⁵Lukas Gonon and Juan-Pablo Ortega. Reservoir computing universality with stochastic inputs. *IEEE transactions on neural networks and learning systems*, 31(1):100–112, 2019.
- ³⁶Daniel J Gauthier, Erik Bollt, Aaron Griffith, and Wendson AS Barbosa. Next generation reservoir computing. *Nature Communications*, 12(5564), 2021.
- ³⁷Thomas L Carroll. Dimension of reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1):013102, 2020.
- ³⁸Thomas L Carroll. Do reservoir computers work best at the edge of chaos? *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(12):121109, 2020.
- ³⁹Florian Stelzer, André Röhm, Kathy Lüdge, and Serhiy Yanchuk. Performance boost of time-delay reservoir computing by non-resonant clock cycle. *Neural Networks*, 124:158–169, 2020.
- ⁴⁰Aaron Griffith, Andrew Pomerance, and Daniel J Gauthier. Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(12):123108, 2019.
- ⁴¹Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, 1996.
- ⁴²Herbert Jaeger et al. *Short term memory in echo state networks*, volume 5. GMD-Forschungszentrum Informationstechnik, 2001.
- ⁴³Chad Nathe, Enrico Del Frate, Thomas Carroll, Louis Pecora, Afroza Shirin, and Francesco Sorrentino. Reservoir computers modal decomposition and optimization. *arXiv preprint arXiv:2101.07219*, 2021.
- ⁴⁴James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN’95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.
- ⁴⁵Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987.
- ⁴⁶Floris Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981.