Naval Research Laboratory
Washington, DC 20375-5320

NRL/6390/MR—2023/2

Computing with Dynamical Systems: Reservoir Computers

THOMAS L. CARROLL

Electromagnetics and Nonlinear Dynamics Section
Material Science and Technology Division

February 9, 2023

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
09-02-2023	NRL Memorandum Report	10/10-2019 – 03/31/2023

4. TITLE AND SUBTITLE

Computing with Dynamical Systems: Reservoir Computers

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Thomas L. Carroll

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
1P99

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/6392/MR--2023/2

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
One Liberty Center
875 N. Randolph Street, Suite 1425
Arlington, VA 22203-1995

10. SPONSOR / MONITOR'S ACRONYM(S)

ONR 6.1 Base

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A reservoir computer is a type of neuromorphic computer that is easy to train. Reservoir computers may be build as highly parallel analog systems, making them potentially useful for edge computing applications, such as unmanned arial vehicles. Obtaining optimal performance from a reservoir computer requires computationally expensive optimization procedures. This work unit applied nonlinear dynamics principles to reservoir computers to establish principles for designing optimal reservoir computers that require less computation than blind optimization.

15. SUBJECT TERMS

Machine learning
Neuromorphic
Artificial intelligence
Synamical systems

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Thomas L. Carroll
U	U	U	U	31	**19b. TELEPHONE NUMBER** *(include area code)* (202) 767-6424

This page intentionally left blank.

I. INTRODUCTION

Reservoir computers were developed as a type of recurrent neural network by machine learning researchers, but they may also be described using the language of dynamical systems. An advantage of reservoir computers over other machine learning techniques is that training a reservoir computer is fast and simple; the reservoir itself, a network of nonlinear nodes, is kept fixed, and the time series responses from the nodes are combined using a linear weighted sum, where the weights are varied to fit a training signal.

Reservoir computers have been shown to be useful for solving a number of problems, including reconstruction and prediction of chaotic attractors, recognizing speech, handwriting or other images or controlling robotic systems. One attractive feature of reservoir computers is that they may be implemented in a wide range of analog hardware, making them potentially very fast but with low power consumption. Examples of reservoir computers so far include photonic systems, analog circuits, mechanical systems and field programmable gate arrays .

There is some conventional wisdom on how to design reservoir computers, but most of that wisdom comes from only one type of reservoir computer, has not actually been tested, and ignores the fact that a reservoir computer is a nonlinear dynamical system. In this project we applied principles from dynamical systems theory to develop guidelines for optimizing reservoir computers.

II. RESERVOIR COMPUTERS

A reservoir computer is a type of recurrent neural network that is particularly easy to train. A typical reservoir computer is created by connecting a set of nonlinear nodes in a network that includes feedback connections. The first reservoir computers used nodes that were modeled on a hyperbolic tangent function or excitable neurons that responded to an input by spiking. Unlike most neural networks, the network connections in a reservoir computer never change. Instead, training to a reservoir computer takes place by fitting the signals from the individual nodes to a training signal, usually by a linear fit. Because the network never changes, reservoir computers can be constructed from analog systems in which it is not possible to alter the connections between nodes.

Manuscript approved February 8, 2023.

Because the connections between nodes do not change, only the output parameters, training a reservoir computer can be faster than training a conventional neural network. Training a reservoir computer that has M nodes requires finding M linear fit coefficients. Train by adjusting network connections requires adjusting at least M^2 parameters, and most implementations of neural networks have far more parameters than this. In addition, stability is a concern in recurrent neural network training, so the number useful node activation functions is limited.

A reservoir computer may be described by

$$\chi_i\left(n+1\right) = f\left(\chi_i\left(n\right)\right) + \sum_{j=1}^{M} A_{ij}\chi_j\left(n\right) + w_i s\left(t\right) \tag{1}$$

where the reservoir computer variables are the $\chi_i(n), i = 1...M$ with M the number of nodes, A is an adjacency matrix that described how the different nodes in the network are connected to each other, $\mathbf{W} = [w_1, w_2, ...w_M]$ describes how the input signal $s(t)$ is coupled into the different nodes, and f is a nonlinear function.

When the reservoir computer was driven with $s(t)$, the first 1000 time steps were discarded as a transient. The next N time steps from each node were combined in a $N \times (M+1)$ matrix

$$\Omega = \begin{bmatrix} \chi_1\left(1\right) & \dots & \chi_M\left(1\right) & 1 \\ \chi_1\left(2\right) & & \chi_M\left(2\right) & 1 \\ \vdots & & \vdots & \vdots \\ \chi_1\left(N\right) & \dots & \chi_M\left(N\right) & 1 \end{bmatrix} \tag{2}$$

The last column of Ω was set to 1 to account for any constant offset in the fit. The training signal is fit by

$$h(t) = \Omega\mathbf{C} \tag{3}$$

where $h(t) = [h\left(1\right), h\left(2\right) \dots h\left(N\right)]$ is the fit to the training signal $g(t) = [g\left(1\right), g\left(2\right) \dots g\left(N\right)]$ and $\mathbf{C} = [c_1, c_2 \dots c_N]$ is the coefficient vector.

The fit coefficient vector is then found by

$$\mathbf{C} = \Omega_{inv} g(t) \tag{4}$$

where Ω_{inv} is the Moore-Penrose pseudo-inverse of Ω and $S^{'}$ is an $(M+1) \times (M+1)$

diagonal matrix constructed from S, where the diagonal element $S'_{i,i} = S_{i,i}/(S^2_{i,i} + k^2)$, where $k = 1 \times 10^{-5}$ is a small number used for ridge regression to prevent overfitting.

The training error may be computed from

$$\Delta_{RC} = \frac{\text{std}\left[\Omega\mathbf{C} - g(t)\right]}{\text{std}\left[g(t)\right]} \tag{5}$$

where std[ ] indicates a standard deviation.

The real measure of reservoir computer performance is how well it can estimate an unknown signal. The reservoir computer is driven with a new input signal $s'(t)$ with the goal of finding the unknown signal $g'(t)$. The matrix of signals from the reservoir is now Ω'. The coefficient vector $\mathbf{C}$ is the same vector we found in the training stage. The testing error is

$$\Delta_{tx} = \frac{\|\Omega'\mathbf{C} - g'\|}{\|g'\|} \tag{6}$$

The testing error measures how accurately the reservoir computer actually solves a problem.

A. Types of Reservoir Computer

Several types of reservoir computer were used in this work. One commonly used reservoir computer is the leaky tanh

$$r_i\left(n+1\right) = \alpha r_i\left(n\right) + \left(1-\alpha\right)\tanh\left(\sum_{j=1}^{M} A_{ij} r_j\left(n\right) + w_i s\left(t\right) + 1\right). \tag{7}$$

Another reservoir computer is described by a simple tanh function

$$\mathbf{R}\left(n+1\right) = g\tanh\left(\mathbf{A}\mathbf{R} + \varepsilon s\left(n\right)\right) \tag{8}$$

We also used a reservoir computer described by a polynomial ordinary differential equation

$$\frac{dr_i\left(t\right)}{dt} \tag{9}$$

$$= \alpha\left[p_1 r_i\left(t\right) + p_2 r_i^2\left(t\right) + p_3 r_i^3\left(t\right) + \sum_{j=1}^{M} A_{ij} r_j\left(t\right) + W_i s\left(t\right)\right]. \tag{10}$$

III. NEW STATISTICS

We modified existing statistics to be able to apply them to reservoir computers. We found these new statistics were useful for characterizing the performance of reservoir computers.

A. Covariance Rank

The rank of the matrix of reservoir computer signals Ω is important for characterizing performance. The individual columns of Ω will be used as a basis to fit the training signal $g(t)$. The columns of Ω may be correlated with each other, so we would like to know the number of uncorrelated columns in Ω.

Principle component analysis states that the eigenvectors of the covariance matrix of Ω, $\Theta = \Omega^T\Omega$, form an uncorrelated basis set. The rank of the covariance matrix tells us the number of uncorrelated vectors. Therefore, we will use the rank of the covariance matrix of Ω,

$$\Gamma = \text{rank}\left(\Omega^T\Omega\right) \tag{11}$$

to characterize the reservoir matrix Ω. We calculate the rank using the MATLAB rank() function, which returns the number of singular values above a certain threshold. The threshold is $\gamma_r = D_{\max}\delta\left(\sigma_{\max}\right)$, where D_{max} is the largest dimension of Ω and $\delta(\sigma_{max})$ is the difference between the largest singular value of Ω and the next largest double precision number. Higher covariance rank usually correlates with lower testing error.

B. Path Length and Coupling Statistics

The length of the path between two nodes in a network is a well known statistic. For an unweighted network, if there is an edge between two nodes, the distance between them is 1. The statistic was modified for the weighted adjacency matrices used in this work. For two nodes i and j, a weighted path distance was defined as

$$\delta_{ij} = \ln\left[\frac{1}{|A_{ij}| + |A_{ji}|}\right]. \tag{12}$$

This form was chosen to be analogous to a distance: if the connection between two nodes was stronger, the weighted distance would be smaller. The statistic is not actually a distance because it can take on negative values. The absolute value was used so that the weighted distance only depended on the overall size of the interaction. In large reservoirs, there were many small values for the weighted distance and a few much larger values, so the natural log was taken to make the variation in the statistic easier to plot.

The weighted path length $L_p(i,j)$ between any two nodes i and j is then found by a breadth-first search. For node i_0, the following algorithm puts the weighted path length to all the other nodes into the vector distanceList:

distanceList$(1 \ldots M) \leftarrow \infty$
distanceList$(i_0) \leftarrow 0$
queue $\leftarrow i_0$
while queue is not empty **do**
  queue2 is empty
  for $k = 1$ to length of queue **do**
    $i_k \leftarrow$ queue(k)
    inList $\leftarrow A_{i_k,1\ldots M}$
    outList $\leftarrow A_{1\ldots M,i_k}$
    nodeList $\leftarrow$ inList $\cup$ outList
    for $j = 1$ to length of nodeList **do**
      $i_M \leftarrow$ nodeList(j)
      if distanceList$(i_M) \neq \infty$ **then**
        add i_M to queue2
        distanceList$(i_M) \leftarrow$ distanceList$(i_k) + \ln\left(\frac{1}{|A_{i_M,i_k}|+|A_{i_k,i_M}|}\right)$
      end if
    end for
  end for
  queue $\leftarrow$ queue2
end while

The weighted path lengths can be negative because of the natural log term, so they are not true distance measurements. If there is no path between two nodes, the weighted path length is ∞ and it is not used in calculating the mean path length. Distances with values of 0 are also not included in the mean weighted path length.

The mean coupling statistic is closely related to the mean weighted path length. The mean coupling statistic is

$$K_m = \frac{1}{M}\sum_{i=1}^{M}\left[\sum_{j=1}^{M}|A_{ij}|\right]. \tag{13}$$

The mean coupling statistic is the mean of the sum of the magnitudes of the incoming

coupling to each node. It is a measure of how tightly the nodes in the reservoir are coupled to each other.

C. Dimension

The rank of a set of signals and the dimension of a set of signals need not be the same. An example is a sine wave; the dimension of the sine wave is one, it requires a two dimensional space to be embedded, and its rank is two, because shifting the sine wave by 90 degrees produces an orthogonal waveform.

The different dimension computation methods described below are measuring different things. The false nearest neighbor dimension measures how much we know about an attractor in $d+1$ dimensions based on knowing the attractor in d dimensions. The covariance dimension estimates the probability, based on the eigenvalues of the covariance matrix, that a d dimensional signal could not have been drawn from a uniform distribution. The false nearest neighbor method characterizes predictability, while the covariance dimension only depends on geometry- the signal being characterized need not be deterministic.

Correlation dimension is a global measurement that estimates the fractal dimension of a signal. It can be difficult to get good measurements of correlation dimension, especially for high dimensional systems, so correlation dimension is not used here. The Kaplan-Yorke dimension is an estimate of the correlation dimension that can be calculated from the Lyapunov exponent spectrum of a chaotic system, so the Kaplan-Yorke dimension is used here in place of the correlation dimension.

1. False Nearest Neighbors Dimension D_{fnn}

To calculate the false nearest neighbor dimension from a single time series, the time series is first embedded in a d dimensional space using the method of delays , and the n nearest neighbors are found. The time series is then embedded in $d+1$ dimensions. The question asked is what fraction of n nearest neighbors in d dimensions are no longer among the n nearest neighbors in $d+1$ dimensions? When the time series requires more than d dimensions for embedding, a large fraction of the nearest neighbors in d dimensions will not be nearest neighbors in $d+1$ dimensions. For some dimension d, the false nearest neighbor fraction

will drop below some arbitrary threshold. This value of d is taken to be the embedding dimension.

For the reservoir computer, rather than a single time series, there is a time series signal for each node. Instead of embedding a signal in progressively higher dimensions, we create a d dimensional signal by using d individual node signals. From eq. (2), the reservoir computer signals may be arranged in an $N \times (M+1)$ matrix Ω, where N is the number of points in each time series and M is the number of nodes. An $N \times d$ dimensional signal $\mathbf{u}$ is created by randomly choosing d of the first M columns from Ω. An index point i_0 is randomly chosen on $\mathbf{u}$ and the 10 nearest neighbors to $\mathbf{u}(i_0)$ are found.

An $N \times (d+1)$ dimensional signal $\mathbf{u}'$ is then created by adding one more randomly chosen column to $\mathbf{u}$. The 10 nearest neighbors to $\mathbf{u}'(i_0)$ are then located. The fraction of false nearest neighbors, $f_{fnn}(i_0, d)$, is the fraction of points that are within the 10 nearest neighbors to $\mathbf{u}(i_0)$ but not $\mathbf{u}'(i_0)$.

The fraction of false nearest neighbors upon going from d to $d+1$ is evaluated for 100 randomly chosen values of i_0 and the mean value , $f_{fnn}\,(u,d) = 1/N_i \sum_{i_0=1}^{N_i} f\,(i_0,d)$, where $N_i = 100$ is calculated. Next, a new random set of columns is chosen from Ω to create a new signal $\mathbf{u}$ and the process is repeated for a new set of 100 randomly chosen values of i_0. Creating the signal $\mathbf{u}$ is repeated 10 times. There are then 10 values for $f_{fnn}\,(u,d)$, each corresponding to a different random choice of d columns.

An arbitrary cutoff value must be chosen for the fraction of false nearest neighbors. If the fraction of false nearest neighbors is above this cutoff, it is assumed that going from d to $d+1$ columns reveals more information about the set of reservoir signals . If the fraction of false nearest neighbors is below this cutoff, then adding more than d columns to the signal $\mathbf{u}$ does not reveal any new information. For this paper, the cutoff was set where the fraction of false neighbors dropped below 0.1. This cutoff is completely arbitrary; the arbitrariness of this cutoff is a problem with the false nearest neighbor method. The false nearest neighbor dimension for each random combination of d columns was $d_{fnn}(u)$. The false nearest neighbor dimension for the reservoir was the mean of the false nearest neighbor dimensions for the $N_u = 10$ vectors $\mathbf{u}$:

$$D_{fnn} = \frac{1}{N_u} \sum_{k=1}^{N_u} d_{fnn}\,(u_k). \tag{14}$$

2. Covariance Dimension D_c

Dimension may also be estimated by whether a d dimensional signal can be distinguished from a Gaussian random signal with d components. The null hypothesis for this method is that the signal of interest comes from a Gaussian random process, and therefore is isotropic in space. We seek to disprove this null hypothesis by embedding the signal in d dimensions and comparing the eigenvalues of the covariance matrix to what would be obtained for a Gaussian random signal.

The covariance matrix for the reservoir was already described in eq. (11). In that example, the covariance matrix was calculated for the entire set of reservoir data. In this section, the goal is to detect anisotropy at the smallest length scales in the reservoir, so the covariance matrix is calculated from small clusters of points.

To understand why small clusters of points are necessary, consider the Lorenz system described above. A plot of the Lorenz signals in two dimensions is clearly anisotropic, even though the Lorenz system is three dimensional.

An $M \times d$ dimensional signal $\mathbf{u}$ is found by randomly selecting d columns from the matrix Ω, as was done for the false nearest neighbor dimension. A small cluster of $d+1$ points is found by picking an index point i_0 and finding the $d+1$ nearest neighbors. Clusters with $d+1$ points are just larger than the smallest clusters useful for calculating a d dimensional covariance matrix. The cluster of $d+1$ points from $\mathbf{u}$ will be called $\mathbf{v}$, so $\mathbf{v}$ has dimensions $(d+1) \times d$. This cluster is then normalized by subtracting the mean from each component and dividing by the standard deviation

$$\mathbf{w}_j = \frac{\mathbf{v}_j - \bar{\mathbf{v}}_j}{\sqrt{\sum_{j=1}^{d} \sum_{i=1}^{d+1} \left[\mathbf{v}_j\left(i\right) - \bar{\mathbf{v}}_j\right]^2}} \tag{15}$$

where the overbar operator indicates the mean and the subscript j indicates one of the d components of the vector $\mathbf{v}$. Next, the $d \times d$ covariance matrix is found

$$C = \frac{\mathbf{w}^T \mathbf{w}}{d+1} \tag{16}$$

A d dimensional Gaussian random process will be isotropic in a d dimensional space. The eigenvalues for a $d \times d$ covariance matrix for a Gaussian random signal n points long are known to converge to a Marchenko-Pastur distribution as n and d approach ∞. For

finite n and d, the eigenvalues for the covariance matrix for a Gaussian random signal may be estimated by drawing random $d \times d$ matrices from the Wishart distribution with a mean covariance matrix equal to the identity. For this paper, the limiting eigenvalues for a Gaussian random process were estimated by creating 10,000 random covariance matrices for each combination of n and d using the MATLAB wishrnd() function.

To study the reservoir signals, a covariance matrix was created from each cluster of points $\mathbf{v}$ as in eqs. (15-16) and the eigenvalues for this matrix were calculated. Once again, 10 random combinations of d columns from Ω were chosen to create $\mathbf{u}$, and for each $\mathbf{u}$ 100 random indices i_0 were used to find $d+1$ nearest neighbors.

For a given signal $\mathbf{u}$, the number of clusters $\mathbf{v}$ with at least one covariance matrix eigenvalue outside the limits for a Gaussian random process was recorded. A total of 100 clusters were created, and the fraction of clusters with an eigenvalue outside the Gaussian random limits was f_{ev}.

The fraction f_{ev} was calculated for values of d from 2 to 100 for a 100 node reservoir. The lowest value of d for which f_{ev} exceeded 0.9 was taken as the covariance dimension $d_c(u)$ for $\mathbf{u}$. A fraction of 0.9 meant that there was a 90% probability that the d dimensional signal $\mathbf{u}$ was not isotropic, and so did not come from a Gaussian random process. The threshold of 0.9 was arbitrary.

The covariance dimension for the reservoir was the mean of the covariance dimensions for the individual combinations $\mathbf{u}$:

$$D_c = \frac{1}{N_u} \sum_{k=1}^{N_u} d_c\left(u_k\right). \tag{17}$$

D. Entropy Statistic

Measuring entropy requires a partitioning of the dynamical system. It was found that the permutation entropy method avoided this coarse graining problem because it creates partitions based on the time ordering of the signals. Each individual node time series $r_i(t)$ was divided into windows of 4 points, and the points within the window were sorted to establish their order; for example, if the points within a window were 0.1, 0.3, -0.1 0.2, the ordering would be 2,4,1,3. Each possible ordering of points in a signal $r_i(t)$ represented a symbol $\psi_i(t)$.

At each time step t, the individual node signals were combined into a reservoir computer symbol $\Lambda(t) = [\psi_1(t), \psi_2(t), \ldots \psi_M(t)]$. With $M = 100$ nodes there were potentially a huge number of possible symbols, but the nodes were all driven by a common drive signal, so only a tiny fraction of the symbol space was actually occupied, on the order of tens of symbols for the entire reservoir computer.

If K total symbols were observed for the reservoir computer for the entire time series, then the reservoir computer entropy was

$$H = -\sum_{k=1}^{K} p\left(\Lambda_k\right) \log\left(p\left(\Lambda_k\right)\right) \tag{18}$$

where $p(\Lambda_k)$ is the probability of the k'th symbol.

E. Memory Statistics

The theory of computation for cellular neural networks states that computational capacity is directly related to the memory capacity of the computer. As a result, many studies measure the memory capacity of reservoir computers. There is a commonly used measure of memory capacity, but it does not take into account that the reservoir is nonlinear, so we also introduced other measures of memory.

1. Standard Measure of Memory Capacity

The standard memory capacity is a measure of how well the reservoir computer can predict previous values of the input signal, with the quality of fit being measured by the cross correlation between the input signal $s(n-\tau)$ and the reservoir computer fit to this signal. To avoid confusing correlations induced by the reservoir computer with correlations in the input signal, the reservoir input signal $s(n)$ is a random noise signal- in this work, the noise is Gaussian with a standard deviation of 1. The memory capacity as a function of delay is calculated as

$$\mathrm{MC}_\tau = \frac{\sum_{n=1}^{N} \left(\left[s\left(n-\tau\right) - \bar{s}\right]\left[h_\tau\left(n\right) - \overline{h_\tau}\right]\right)^2}{\sum_{n=1}^{N} \left[s\left(n-\tau\right) - \bar{s}\right]^2 \sum_{n=1}^{N} \left[h_\tau\left(n\right) - \overline{h_\tau}\right]^2} \tag{19}$$

with N is the number of time series points and the overbar indicator indicates the mean. The signal $h_\tau(n)$ is the fit of the reservoir signals $r_i(n)$ to the delayed input signal $s(n-\tau)$.

The total memory capacity is

$$\mathrm{MC} = \sum_{\tau=1}^{\tau_{\max}} \mathrm{MC}_\tau \tag{20}$$

where τ_{max} was set to 100 because at that value MC_τ was small.

There are some drawbacks to this definition of memory. Changing the input signal for a nonlinear system will change its properties- even changing the amplitude of the random input signal will change the effect of the nonlinearities in the reservoir computer, so the memory capacity as defined in eq. (20) may not be a true reflection of the memory of the reservoir computer. Also, the memory capacity calculation requires that one fit a signal, but memory capacity is often used as an independent metric to define how well the reservoir computer will fit signals. Characteristics of the reservoir computer that are independent of the memory but lead it to be better or worse at reproducing signals can affect the memory capacity estimate. Because of these drawbacks, two alternate estimates of memory are used in this paper.

2. Norm of the Variation

Some of the original definitions of memory in dynamical systems are similar to the concepts used to define Lyapunov exponents. Following this path, the variational equation may be used to measure the amount of memory in a reservoir computer. If the initial perturbation is δ_0, the perturbation at n steps later for a nonlinear system is,

$$\delta_n = \left[\prod_{j=0}^{n} D_r f\left(\mathbf{r}\left(j\right)\right)\right] \delta_0 \tag{21}$$

where $f(\mathbf{R})$ is the node nonlinearity and $D_f f(\mathbf{R})$ is the derivative of f with respect to the node variables.

To create a statistic based on the variational equation, for a system with M nodes I created a random $M \times M$ matrix for δ_0 and then made the rows orthonormal. I propagated this matrix with the nonlinear variational equation. I propagate the variation along a known trajectory of the reservoir computer and label the initial variation with j to indicate the particular initial condition. I then calculated the norm of $\delta_n(j)$ after n steps.

I then take the sums

$$\delta_{\text{sum}} = \sum_{n=1}^{\tau_{\max}} \|\delta_n(j)\| \tag{22}$$

where the $\| \ \|$ operator returns the 2-norm of a matrix. The norm of the variation, D_{var}, is the mean of the individual sums

$$D_{var} = \frac{1}{100} \sum_{j=1}^{100} \delta_{\text{sum}}(j) \tag{23}$$

While this statistic, which I call the norm of the variation, is similar to the largest Lyapunov exponent, it does not follow exactly the same pattern. At short delays, the decay of the variation will be dominated by the most negative Lyapunov exponents, while the less negative exponents will govern the decay at later times.

While the norm of the variation does not depend on signal fitting and uses the actual signal driving the reservoir, it can only be found if one has the equations defining the reservoir. The delay capacity method described in the next section can be used when only experimental data is available.

3. Delay Capacity

The delay capacity statistic is adapted from the consistency capacity developed in a paper by Jüngling et al (2021). To estimate the delay capacity I compared the reservoir computer signals at two different times. I whiten both of these sets of signals, calculate the cross covariance and find the trace. Again, to be consistent with the definition of memory capacity I sum the capacities for all delays and divide by the number of delays.

The signals from a reservoir with M nodes and N time series points may be arranged in an $M \times N$ matrix $\mathbf{R}_0$

$$\mathbf{R}_0 = \begin{bmatrix} r_1(1) - \bar{r}_1 & \cdots & r_1(N) - \bar{r}_1 \\ r_2(1) - \bar{r}_2 & & r_2(N) - \bar{r}_2 \\ \vdots & & \vdots \\ r_M(1) - \bar{r}_M & \cdots & r_M(N) - \bar{r}_M \end{bmatrix} \tag{24}$$

where $\bar{r}_i$ is the mean of r_i.

The covariance matrix is then formed as:

$$\mathbf{C} = \frac{\mathbf{R}_0(t)\mathbf{R}_0^T(t)}{N} + L_{reg}\mathbf{I}. \tag{25}$$

The regularization factor $L_{reg} = 10^{-10}$ is added because the covariance matrix can be near singular.

The covariance matrix is then decomposed by a singular value decomposition, $\mathbf{C} = \mathbf{USV^T}$ and the reservoir signals are normalized as

$$\widetilde{\mathbf{R}}_0 = \left(\mathbf{V}^T\mathbf{R}_0\right)\left(\sqrt{\mathbf{S}}\right)^{-1} \tag{26}$$

As a measure of memory, a second matrix is created, $\mathbf{R}_\tau(t) = \mathbf{R}_0(t-\tau)$ and normalized in the same manner to produce $\tilde{\mathbf{R}}_\tau(t)$. The cross covariance between the normalized versions of the regular and delayed matrices is then found

$$\mathbf{C}(\tau) = \frac{\tilde{\mathbf{R}}_0\tilde{\mathbf{R}}_\tau^T}{N}. \tag{27}$$

The delay capacity statistic is calculated as

$$\Theta_d = \frac{\sum_{\tau=0}^{\tau_{\max}} \text{Trace}\left|\mathbf{C}(\tau)\right|}{\tau_{\max}} \tag{28}$$

where the $|\,|$ operator indicates an absolute value. The trace of the cross covariance matrix $\mathbf{C}(\tau)$ is shown in figure **??**.

IV. INPUT SIGNALS

We generated input and training signals from the Lorenz chaotic systemThe first system we used to generate input and training signals is the Lorenz system

$$\begin{aligned} \tfrac{dx}{dt} &= c_1 y - c_1 x \\ \tfrac{dy}{dt} &= x(c_2 - z) - y \\ \tfrac{dz}{dt} &= xy - c_3 z \end{aligned} \tag{29}$$

with c_1=10, c_2=28, and c_3=8/3. The equations were numerically integrated with a time step of $t_s = 0.02$.

V. SYMMETRY

In order to provide a good fit to training and testing signals the reservoir must produce a diverse set of signals. One way to make sure the signals are diverse is to avoid symmetries

in the adjacency matrix **A**. Here we use the concept of symmetry from graph theory, where a symmetry is a permutation of the nodes of the network along with the edges attached to the nodes which leave the network unchanged.

Figure 1 shows the training error as a function of the number of symmetries ζ_s in the adjacency matrix for a leaky tanh reservoir computer driven by the Lorenz system. It can be seen that when the adjacency matrix contains more symmetries, the training error is larger.

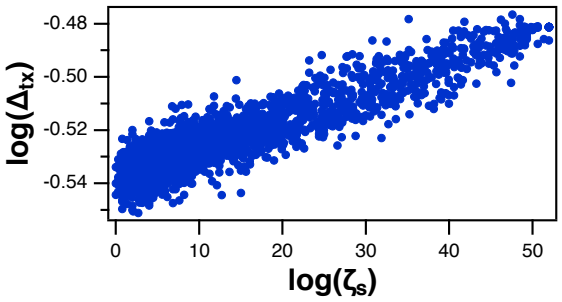

FIG. 1: Reservoir computer testing error Δ_{tx} vs. number of symmetries ζ_s in the network for a reservoir computer with leaky tanh nodes when input signal $s(t)$ is the Lorenz x signal and the training signal $g(t)$ is the Lorenz z signal. The logarithms are base 10.

VI. SIGNAL CLASSIFICATION

In equation (4), the vector of coefficients **C** used to fit the reservoir output to a training signal was defined. This fit coefficient vector may be used as a feature vector for a complex time series.

A reservoir may be driven with two signals, signal i and signal j. The reservoir is trained to predict signals i or j one time step into the future, yielding coefficient vectors $\mathbf{C}_i$ and $\mathbf{C}_j$. The difference between signals i and j is computed as

$$\Delta_{ij} = \sum_{k=1}^{M+1} \sqrt{\mathbf{C}_i^2(k) - \mathbf{C}_j^2(k)} \tag{30}$$

where $\mathbf{C}_i{}^2(k)$ is the $k'th$ component of the coefficient vector for signal i.

As test signals for classification, four signals from the set of Sprott chaotic systems were used. Different reservoir parameters were varied and the error E_C in identifying which of the

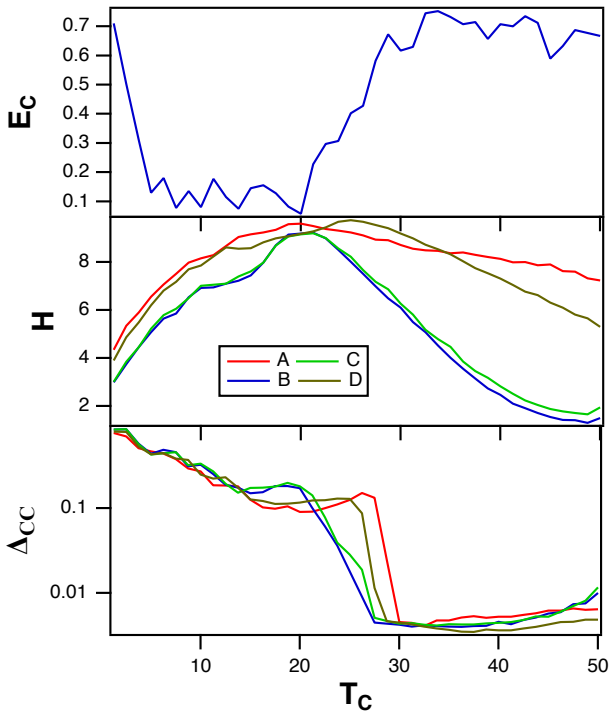

FIG. 2: The top plot shows the fraction of errors E_C in identifying which of the four Sprott systems was present, as a function of the parameter T_C a reservoir computer. The middle plot shows the entropies H for the four Sprott systems over the same parameter range. The bottom plot is the training error Δ_{CC}.

four signals was present was tabulated. Figure 2 shows results found by varying a parameter T_C for a spiking reservoir computer based on the Fitzhugh-Nagumo equations, but these results were typical for other reservoir computers. What we found was that the reservoir entropy H, as defined in Section III D, was important for predicting the classification error E_C. These results were typical for other reservoirs and other types of signals. The bottom plot in fig. 2 is related to the training error Δ_{RC}, the error in actually fitting the training signal. Note that the smallest classification errors do not come for the parameter for which the fit to the training signal is best; rather, it is the entropy that is important.

VII. RESERVOIR COMPUTER DIMENSIONS

Dimension measurements from Section III C 1 and III C 2 reveal that surprisingly, the dimension of the set of reservoir signals is much lower than the dimension of the space

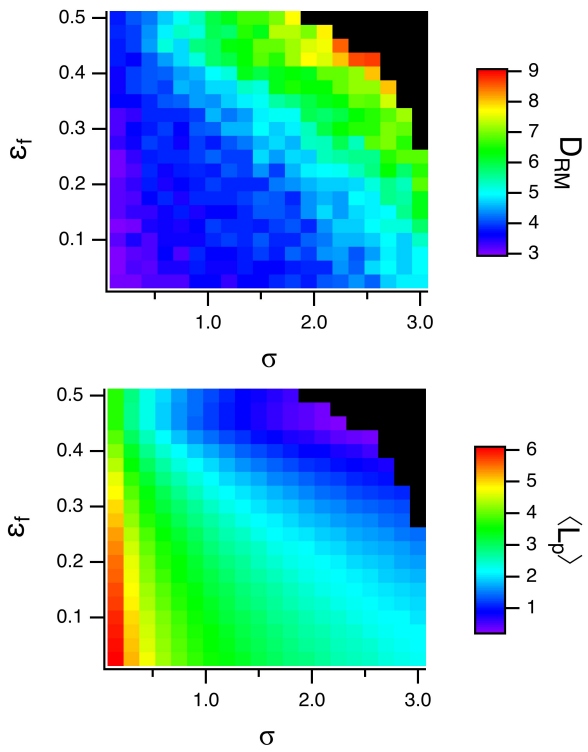

FIG. 3: The top plot shows the reservoir computer manifold dimension D_{RM}. The variable σ is the adjacency matrix spectral radius, while ε_f is a measure of how disordered the adjacency matrix is, where larger ε_f means less order. The bottom plot shows the mean weighted path length $< L_p >$ Smaller values of the mean weighted path length indicate larger interactions between nodes. The input signal came from a chaotic map with an embedding dimension of 7. The black regions indicate where the reservoir computer became unstable.

occupied by these variables. The potential reservoir computer dimension is as high as the number of nodes, which can be 100 or more, but the measured dimension of the set of reservoir computer variables is never greater than six to ten. In hindsight these result is less surprising; the reservoir computer nodes are all driven by a common signal, so they will maintain some relation to each other.

The dimension measured by the covariance dimension will be referred to as the manifold dimension D_{RM}.

Figure 3 shows the reservoir computer manifold dimension and the mean weighted path length statistic from Section III B as the spectral radius σ and the disorder ε_f of the adjacency matrix are changed. Larger values of ε_f indicate more disorder. Figure 3 shows that

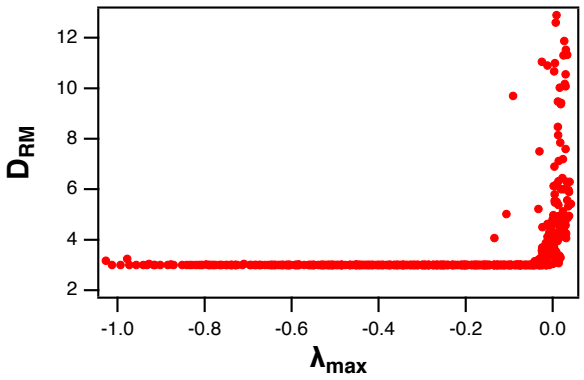

FIG. 4: Reservoir manifold dimension D_{RM} versus the maximum Lyapunov exponent for the reservoir, λ_{max}.

the manifold dimension increases as the spectral radius and the disorder in the adjacency matrix increase. The mean weighted path length decreases as the manifold dimension increases. Smaller mean weighted path length indicates a stronger interaction between nodes, demonstrating that the manifold dimension increases as the interaction between nodes becomes stronger.

Figure 4 in particular shows why the reservoir computer manifold dimension increases. Fig. 4 shows the manifold dimension as a function of the maximum Lyapunov exponent for the reservoir computer. The manifold dimension is small until the Lyapunov exponent approaches zero. Large manifold dimensions occur because the reservoir is near the border of stability.

VIII. EDGE OF CHAOS

One persistent claim about reservoir computers is that their best performance comes at the edge of chaos, the parameter value at which the behavior transitions from ordered to disordered. This claim originated with computational theory of cellular neural networks. Reservoir computers, on the other hand, are dynamical systems, so there are effects in reservoir computers that do not show up in other computational systems.

The behavior of a reservoir near the edge of chaos (more accurately, the edge of stability) was tested with a reservoir computer that was defined by a polynomial ordinary differential equation (eq. 9).

The behavior of this reservoir computer as its largest Lyapunov exponent approaches zero from below depends on the particular parameters.

Figure 5 shows from top to bottom the testing error Δ_{tx}, the maximum Lyapunov exponent for the reservoir computer (λ_{max}) and the information entropy for the reservoir computer. The parameters for this figure were $p_2 = -0.871984$, $p_3 = 0.52492$, the spectral radius σ was 0.28512 and α was 5.53275. The middle plot in figure 5 also shows the maximum of the local Lyapunov exponent for the reservoir computer. The local Lyapunov exponent was calculated over a period of one time step. The positive local Lyapunov exponent caused the reservoir computer became unstable before the global Lyapunov exponent became positive. If the reservoir computer variables enter a region of large positive local Lyapunov exponent, the variables may burst to a large amplitude from which the reservoir computer does not recover, making the reservoir computer unstable even though the global Lyapunov exponent is negative.

In figure 5, the smallest testing error comes just before the reservoir computer becomes unstable as p_1 increases. As the reservoir computer comes closer to the edge of stability, the entropy H increases. This behavior fits the expected pattern where the best computational performance comes at the edge of stability.

Figure 6 shows a different behavior. In figure 6, the smallest training error is not at the edge of stability. In figure 6, the parameters were $p_2 = -1.03594$, $p_3 = 0.9308149$, the spectral radius σ was 2.78752 and α was 2.72261. Once again, positive local Lyapunov exponents (green dot-dash line) cause the reservoir computer to become unstable while the global Lyapunov exponent is still negative.

A. Fractal Dimension

Figures 5 and 6 show different behaviors because the interaction between the Lyapunov exponents of the reservoir and the Lyapunov exponents of the input Lorenz system. The Kaplan-Yorke dimension is an estimate of the capacity (or fractal) dimension based on the spectrum of Lyapunov exponents of a chaotic system. For a spectrum of Lyapunov exponents $\lambda_1 \geq \lambda_2 ... \geq \lambda_d$, the Kaplan-Yorke (or Lyapunov) dimension is

$$D_{KY} = j + \sum_{k=1}^{j} \frac{\lambda_k}{|\lambda_{j+1}|} \tag{31}$$

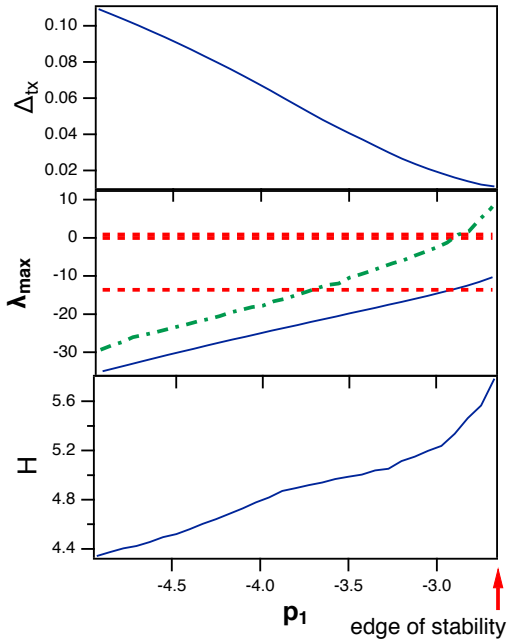

FIG. 5: From top to bottom, the testing error Δ_{tx}, the maximum Lyapunov exponent λ_{max} and the entropy H for the polynomial ODE reservoir computer when the input signal was the Lorenz x signal and the training signal was the Lorenz z signal. The parameter p_1 was varied, while the other parameters were $p_2 = -0.871984$, $p_3 = 0.52492$, the spectral radius σ was 0.28512 and α was 5.53275. The dashed red lines in the middle graph are the Lyapunov exponents for the Lorenz system, while the green dot-dash line is the largest local Lyapunov exponent for the reservoir computer. The positive local Lyapunov exponents caused the reservoir computer became unstable for values of p_1 greater than those shown in the plot. When the reservoir computer became unstable, the values of the reservoir variables diverged to $\pm\infty$.

where j is the largest integer for which the cumulative sum of the Lyapunov exponents is greater than 0 and the $|\ |$ operator indicates the absolute value.

In figure 6, the maximum Lyapunov exponent for the reservoir computer lies in between the Lyapunov exponents for the Lorenz system, leading to an increase in the Kaplan-Yorke dimension for the reservoir computer. The Kaplan-Yorke dimensions for the parameter configurations in both figures 5 and 6 are shown in figure 7.

The Kaplan-Yorke dimension for the Lorenz system used in this work is 2.06. Figure 7 shows that when the smallest testing error comes at the edge of stability, there is only a very small increase in the Kaplan-Yorke dimension of the reservoir signals, while the Kaplan-

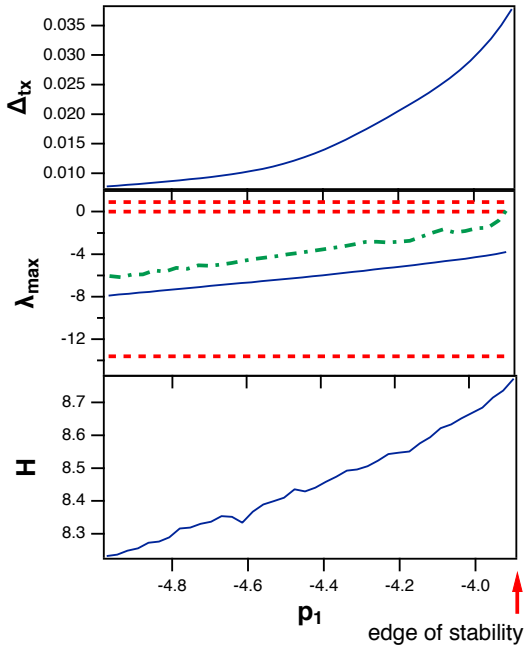

FIG. 6: From top to bottom, the testing error Δ_{tx}, the maximum Lyapunov exponent λ_{max} and the entropy H for the polynomial ODE reservoir computer when the input signal was the Lorenz x signal and the training signal was the Lorenz z signal. The parameter p_1 was varied, while the other parameters were $p_2 = -1.03594$, $p_3 = 0.9308149$, the spectral radius σ was 2.78752 and α was 2.72261. The dashed red lines in the middle graph are the Lyapunov exponents for the Lorenz system, while the green dot-dash line is the largest local Lyapunov exponent for the reservoir computer. The positive local Lyapunov exponents caused the reservoir computer became unstable for values of p_1 greater than those shown in the plot. When the reservoir computer became unstable, the values of the reservoir variables diverged to $\pm\infty$.

Yorke dimension shows a larger increase when the smallest testing error does not come at the edge of stability. Because the dimension of the reservoir computer signals differs from the dimension of the Lorenz system, the reservoir computer is not an embedding of the driving system, leading to a degradation in performance.

The "edge of chaos" rule continues to be popular because most numerical work on reservoir computers uses only a few particular node types. When a larger set of reservoir computers is studied, it can be seen that the "edge of chaos" rule is not always true.

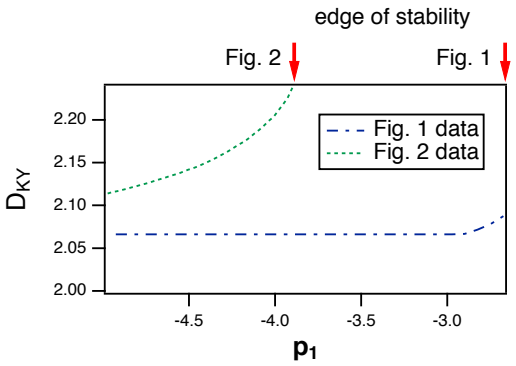

FIG. 7: Kaplan-Yorke dimension D_{KY} as a function of p_1 for the polynomial ODE reservoir computer driven by the Lorenz x signal for the parameter configurations in figures 5 and 5. "Fig. 1" refers to the parameters in figure 5, where the best performance came at the edge of stability, while "Fig. 2" refers to the parameters in figure 6, where the best performance was not at the edge of stability.

IX. OPTIMIZING MEMORY

As the previous section (Section VIII) demonstrated, conventional wisdom for computational systems does not always apply to reservoir computers. Conventional wisdom states that increasing memory capacity should lead to increased computational capacity. The testing error for fitting the Lorenz z signal is using a tanh reservoir (eq 8) is plotted in figure 8.

The three different memory statistics or the Lorenz system are also shown in figure 8. The delay capacity Θ_d in figure 8 roughly echos the memory capacity MC , but because the delay capacity depends on the actual input signal, it is not the same as the memory capacity. The delay capacity is a measure of how long slowly the autocorrelation of signals in a reservoir computer drops off with time, while memory capacity is measured by how well the reservoir computer fits a delayed noise signal. The delay capacity is affected by the autocorrelation of the input signal.

The variation of the norm in figure 8 does more closely resemble the plot of memory capacity. The variation of the norm measures how quickly a perturbation to the reservoir trajectory decays, but unlike the memory capacity, it depends on the signal that drives the reservoir. The decay of a perturbation would seem to be an excellent way to quantify fading

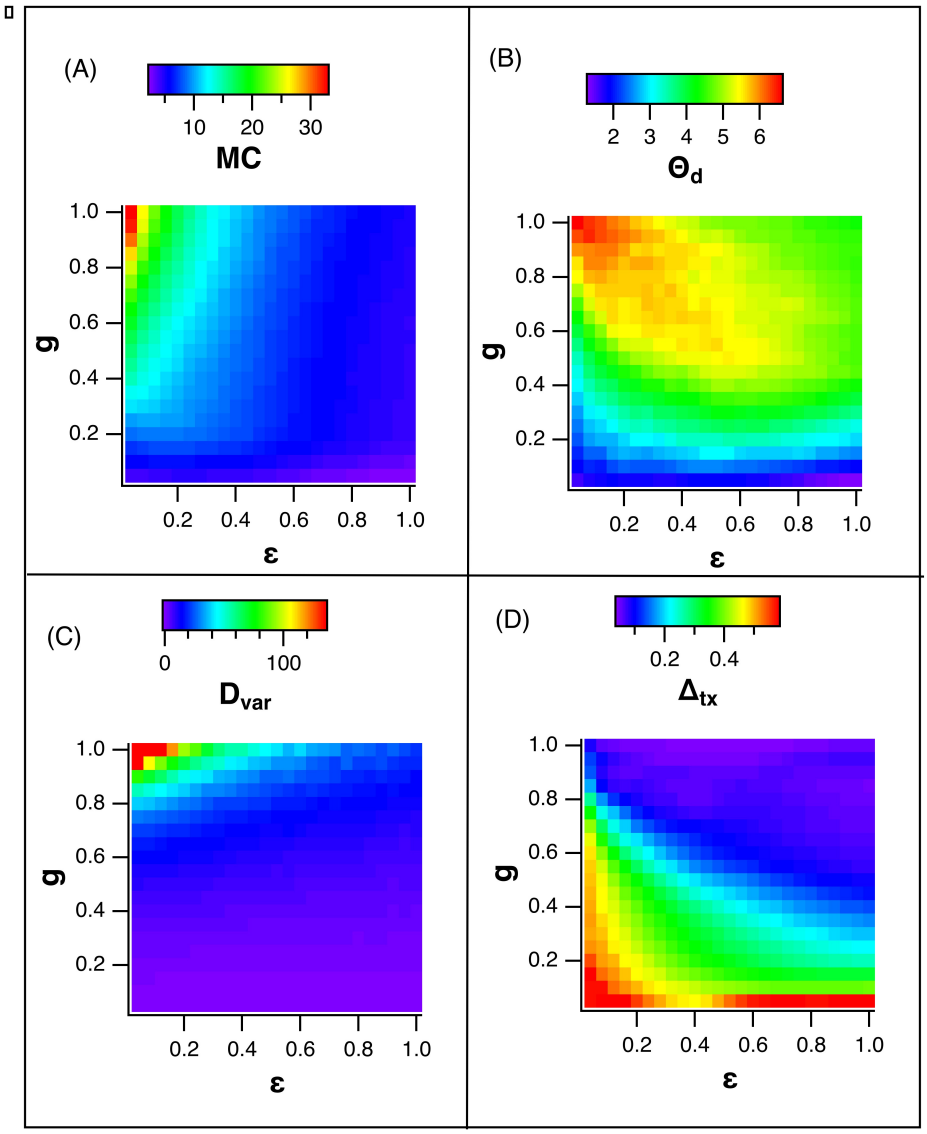

FIG. 8: Testing error and memory statistics for the tanh reservoir (eq 8) driven by the Lorenz x signal and trained on the Lorenz z signal. (A) is the memory capacity MC, (B) is the delay capacity Θ_d as defined in eq. (28) , (C) is the variation of the norm D_{var} (eq (23)) while (D) is the testing error Δ_{tx}. For these parameter ranges the lowest testing error does not occur where the memory capacity is largest

memory, but calculating the norm of the variation requires that the reservoir computer equations are known, which may not be true in an experiment.

The testing error for the Lorenz system is not minimized at the largest values of the memory capacity, shown in figure 8. The different memory statistics in figure 8 are all

maximized for large values of the feedback constant g but small values of the input multiplier ε.

A. Variable Memory Reservoir

A reservoir computer with variable memory may be created by adding extra dimensions to the tanh reservoir of eq. (8). The variable memory reservoir is

$$\begin{aligned} r_{i,1}(n+1) &= g \tanh\left(\sum_{j=1}^{M} A_{i,j} r_{i,1}(n) + \varepsilon s(n) + 0.5 r_{i,d_e}\right) \\ r_{i,j}(n+1) &= r_{i,j-1}(n) \quad j = 2 \ldots d_e \end{aligned} \tag{32}$$

where d_e is the dimension of the node activation function and $r_{i,j}(n)$ is the jth component of the i'th node. One could create similar nodes for an ordinary differential equation system by using a delayed signal.

Figure 9 shows the three different measures of memory used in this work for the multidimensional tanh nodes of eq. (32) as the node dimension d_e was scanned and the parameter g was fixed at 0.35, $\varepsilon = 0.5$ and the spectral radius $\rho = 1$. These parameter values were set by scanning through a number of parameter values and node dimensions and choosing the values of g and ε that gave a minimum testing error for a range of d_e between 5 and 10. The input signal for all three statistics was a Lorenz x signal. To make it possible to plot all three statistics on one axis, all statistics were normalized by their maximum values. The memory variations were similar when the input signal was the Rössler x signal.

All three memory statistics in figure 9 are consistent, and all three show that increasing the node dimension d_e increased the memory for the reservoir computer. It was also found that changing the node dimension d_e did not change the largest Lyapunov exponent for the reservoir.

To demonstrate the interaction between memory testing error, a variable order nonlinear autoregressive moving average (NARMA) system was used as an input to the multidimensional reservoir computer. The NARMA system is

$$y(n+1) = 0.3y(n) + 0.05y(n)\sum_{j=1}^{N_N} y(n-j) + 1.5u(n-N_N+1)\,u(n) + 0.1 \tag{33}$$

where the order of the model is N_N. The input signal $u(n)$ is drawn from a uniform random distribution between 0 and 0.5.

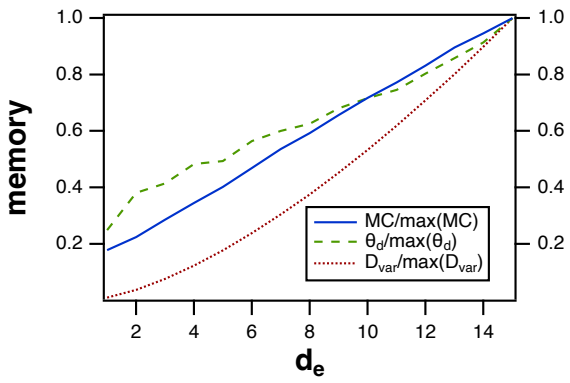

FIG. 9: The memory capacity MC, the delay capacity θ_d and the norm of the variation D_{var} for the reservoir computer with multidimensional nodes. All statistics were normalized by their maximum values so the could be plotted on one axis. The input signal was the Lorenz x signal.

The dependance of the testing error on both memory capacity and the memory required to reproduce a particular signal may be investigated by using NARMA systems of varying orders. Equation 33 described a NARMA system of order N_N; in this section, N_N varies from 1 to 10.

Figure 10 shows the testing error when both the node dimension d_e and the NARMA order N_N are varied. For this plot, $g = 0.35$ and $\varepsilon = 0.35$ were chosen by varying both of these parameters for a NARMA system with $N_N = 10$ and choosing the values that gave the lowest training error.

For NARMA systems with N_N from 1 to 4, the lowest training error occurred for $d_e = 1$. For higher order NARMA systems, the lowest training error was seen when $N_d = d_e - 1$, demonstrating that the lowest training error came when the memory capacity for the reservoir matched the memory required by the problem being solved.

X. EXPANDING RESERVOIR COMPUTERS

The reservoir covariance rank, as described in Section III A, was a useful predictor of reservoir computer performance. Figure 11 shows the mean testing error Δ_{tx} as a function of covariance rank Γ for six combinations of input signal and reservoir computer.

In Fig. 11 the mean testing error depends on the covariance rank Γ, not on the number of nodes in the reservoir computer. The tanh data has twice the rank because the set of

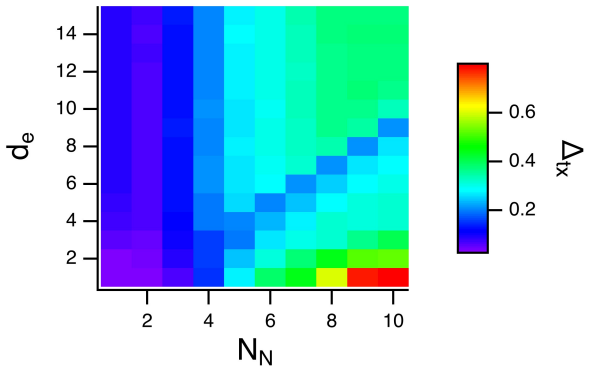

FIG. 10: Testing error for the multidimensional tanh nodes of eq. (32) for the NARMA system of variable order N_N as the node dimension d_e varied. This figure shows that the optimal memory capacity for producing a small testing error in this NARMA system depends on the order of the system.

reservoir computer signals $r_i(t)$ were supplemented with their squares $r_i^2(t)$. Using more nodes allows the covariance rank to be larger, since it is limited by the size of the reservoir. For a particular rank there is a spread in testing error because there are other factors that affect the error, such as the reservoir Lyapunov exponents, but the main effect of the number of nodes is to increase the possible rank. The point of Fig. 11 is that the curve of testing error versus rank does not depend on the number of nodes in the reservoir computer, but rather on the covariance rank.

Figure 11 does show that for the highest ranks in the laser system the testing error actually increases. The laser nonlinearity is bounded, so for some parameter combinations it is possible the laser equation has one or more positive Lyapunov exponents. Positive Lyapunov exponents would lead to complex chaotic signals that had high rank, but they would also cause the breakdown of generalized synchronization, so the testing error would be large.

The rank of the reservoir can be increased by adding extra delayed versions of the reservoir signals. The matrix of signals from the reservoir computer was Ω_1 as in Eq. (2). The time-shifted reservoir was

$$\Omega_2(i,j) = \nu\left(k\theta + (i-1)\tau_D - \tau_j\right) \tag{34}$$

where $k = \mathrm{mod}(j, M_1)$, $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M_2$

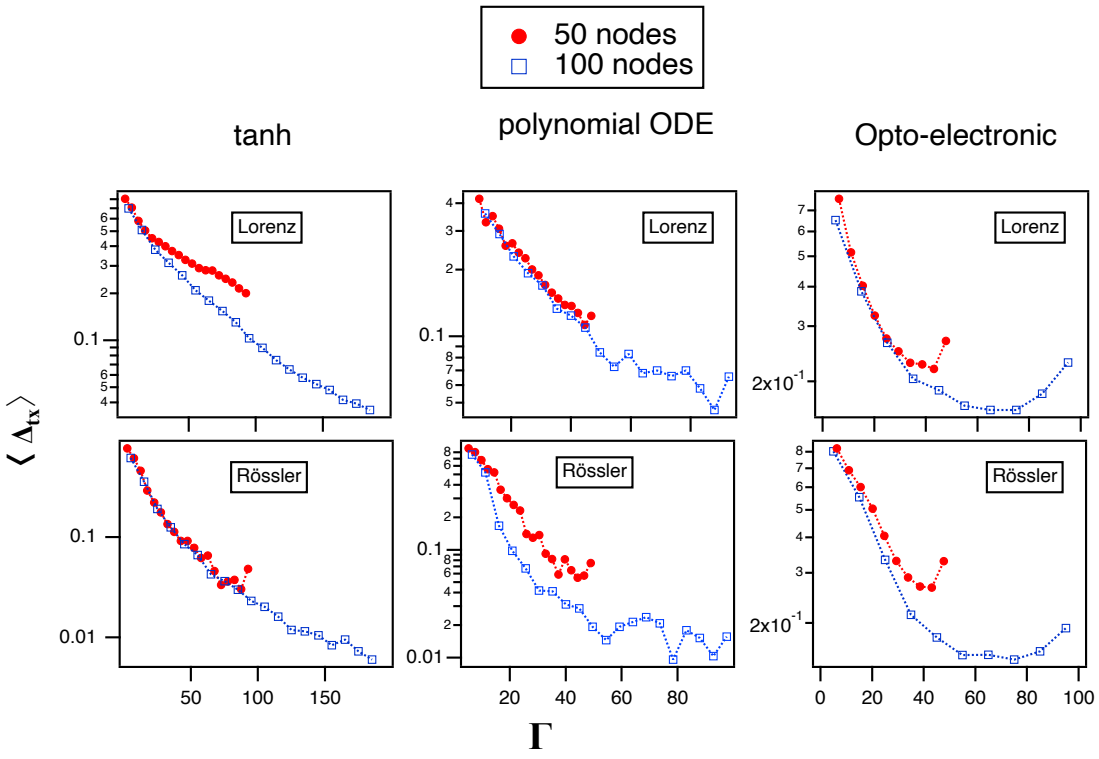

FIG. 11: Mean testing error $\langle \Delta_{tx} \rangle$ as a function of covariance rank Γ for Lorenz or Rössler x signals driving either a tanh, a polynomial ODE or a opto-electronic reservoir computer. The reservoir computers were trained on the z signals from the corresponding input systems. Reservoir computers containing 50 or 100 nodes were used. The tanh data has twice the rank because the set of reservoir computer signals $r_i(t)$ were supplemented with their squares $r_i^2(t)$.

The maximum time shift value was τ_{max}, while the individual time shifts were $\tau_j = j \times \tau_{max}/M_2$. When the time shift τ_j was not an integer multiple of θ, linear interpolation was performed between the two nearest θ-sampled values of the opto-electronic reservoir computer output signal ν to get the shifted values. We simulated the testing error as τ_{max} was varied with $M_1 = M_2 = 100$ nodes and picked a value that gave a small testing error. The reservoir could also be expanded by appending a matrix of reservoir signals passed through finite impulse response filters.

To find out how many actual nodes are needed for a reservoir computer, the laser delay reservoir computer was simulated for a variety of reservoir sizes $M_1 = 2, 3, \ldots M_2$ and with a fixed size M_2 for the time-shifted matrix. The reservoir was driven with the Lorenz x signal and fit the Lorenz z signal. The maximum delay value τ_{max} was $10\tau_D$, which was equivalent

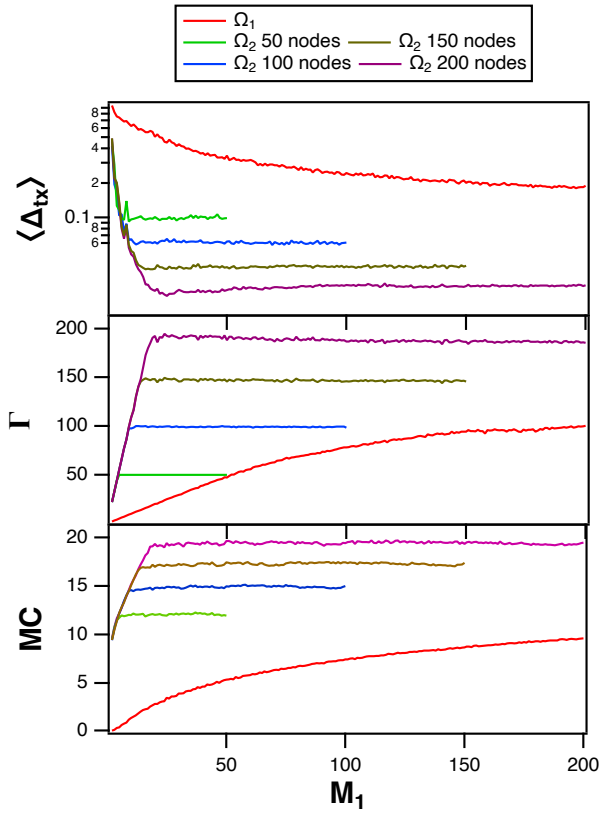

FIG. 12: The top plot is the mean testing error for the opto-electronic reservoir computer when the input signal is the Lorenz x signal and the training signal is the z signal. The trace labeled Ω_1 is the error for the original reservoir computer with M_1 nodes. The traces labeled as Ω_2 are for the time-shifted matrix with different fixed values of M_2. The middle plot shows the covariance rank Γ while the bottom plot shows the memory capacity MC.

to 10 Lorenz time steps. Figure 12 shows the training error, covariance rank and memory capacity based on either Ω_1 or Ω_2 as M_1 varies. The elements of the input vector $\mathbf{W}$ were randomly assigned to -1 or 1. Because the training and testing error can depend on the input vector, 20 different realizations were performed, each with a different input mask, and all reported results are the mean values over these 20 realizations.

In Fig. 12, the covariance rank for the reservoir output matrix Ω_1 increases with the number of nodes, but the increase slows as the number of nodes increases. Except for the case where the time-shifted reservoir matrix contains $M_2 = 50$ nodes, the covariance rank of the shifted reservoir matrix is always higher than the covariance rank of the unshifted matrix Ω_1. The rank of the time-shifted reservoir saturates when M_1 is approximately equal

to $M_2/10$. The time-shifted reservoir matrices also have a larger memory capacity than the unshifted reservoir matrix.

Figure 12 also shows that the testing error for the time-shifted reservoir computer is always lower than the error for the unshifted version, and using the shifted signals allows a small testing error with a reservoir of only a few nodes. These results were confirmed with an experiment with an opto-electronic delay system.

XI. PUBLICATIONS

T. L. Carroll, "Using reservoir computers to distinguish chaotic signals", *Physical Review E*, vol. 98, pp. 052209, 5. 2018

T. L. Carroll and L. M. Pecora, "Network structure effects in reservoir computers", *Chaos*, vol. 29, pp. 083130, 8. 2019

T. L. Carroll, "Dimension of reservoir computers", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, pp. 013102, 1. 2020

T. L. Carroll, "Path length statistics in reservoir computers", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, pp. 083130, 8. 2020

T. L. Carroll, "Do reservoir computers work best at the edge of chaos?", *Chaos*, vol. 30, pp. 121109, 12. 2020

T. L. Carroll, "Adding filters to improve reservoir computer performance", *Physica D: Nonlinear Phenomena*, vol. 416, pp. 132798, 2021

T. L. Carroll, "Low dimensional manifolds in reservoir computers", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, pp. 043113, 4. 2021

T. L. Carroll, "Optimizing Reservoir Computers for Signal Classification", *Frontiers in Physiology*, vol. 12, pp. 893. 2021

T. L. Carroll, "Optimizing memory in reservoir computers", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, pp. 023123, 2. 2022

T. L. Carroll, "Creating New Chaotic Signals with Reservoir Computers", *Chaos, Solitons & Fractals*, vol. in press, pp. 2022

T. L. Carroll and J. D. Hart, "Time shifts to reduce the size of reservoir computers", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, pp. 083122, 8. 2022

XII. INVITED TALKS

T. L. Carroll and L. M. Pecora, Network Structure and Reservoir Computers, SIAM Conference on Applications of Dynamical Systems, Snowbird, Utah May 21 2019

T. L. Carroll, Nonlinear Dynamics of Reservoir Computers, Illinois State University virtual talk, October 16, 2020

T. L. Carroll, Do Reservoir Computers Work best at the Edge of Chaos?, SIAM Conference on Applications of Dynamical Systems (virtual), May 25 2021

T. L. Carroll, Choosing Optimal Reservoir Computers, APS March Meeting (live) 16 March 2022 Chicago IL.

T. L. Carroll, Time Shifts to Reduce the Size of Reservoir Computers, Third Symposium on Machine Learning and Dynamical Systems, 26-30 September 2022, Toronto, Canada

XIII. CONTRIBUTED TALKS

T. L. Carroll and L. M. Pecora, ?Network Structure Effects in Reservoir Computers?, Department of Defense Artificial Intelligence/Machine Learning Technical Exchange Meeting, 29 October 2019, Arlington, VA

T. L. Carroll and L. M. Pecora, ?Network Structure Effects in Reservoir Computers?, Dynamics Days 2020, 3 January 2020, Hartford, CT

T. L. Carroll, Network Statistics for Reservoir Computers, Second Symposium on Machine Learning and Dynamical Systems (Virtual) September 2020 (Fields Institute, Toronto)

T. L. Carroll, Do Reservoir Computers Work best at the Edge of Chaos?, Dynamics Days 2022 (virtual) 8 January 2022.

T. L. Carroll, Choosing Optimal Reservoir Computers, Dynamics Days 2023 (virtual) 10 January 2023