# Global Forecasts in Reservoir Computers

S. Harding,[1, a)] Q. Leishman,[2, b)] W. Lunceford,[2, c)] D. J. Passey,[3, d)] T. Pool,[4, e)] and B. Webb[2, f)]
[1)]*Brigham Young University, Provo, UT 84602, USA*
[2)]*Brigham Young University, Provo, UT 84604, USA*
[3)]*University of North Carolina at Chapel Hill, NC 27599, USA*
[4)]*Carnegie Mellon University, Pittsburg, PA 15289, USA*

(Dated: 18 July 2024)

Abstract: A reservoir computer is a machine learning model that can be used to predict the future state(s) of time-dependent processes, e.g. dynamical systems. In practice, data, in the form of a an input-signal is fed into the reservoir. The trained reservoir is then used to predict the future state of this signal. We develop a new method for not only predicting the future dynamics of the input-signal but also the future dynamics starting at an arbitrary initial condition of a system. The systems we consider are the Lorenz, Rossler, and Thomas systems restricted to their attractors. This method, which creates a global forecast, still uses only a single input-signal to train the reservoir but breaks the signal into many smaller windowed signals. We examine how well this windowed method is able to forecast the dynamics of a system starting at an arbitrary point on a system's attractor and compare this to the standard method without windows. We find that the standard method has almost no ability to forecast anything but the original input-signal while the windowed method can capture the dynamics starting at most points on an attractor with significant accuracy.

**Historically, reservoir computers have been used to forecast the future state of dynamical systems by training the reservoir on a single finite trajectory of the system. While some features of the system can be deduced from these reservoirs, the ability to create a global forecast using such reservoirs has been out of reach. Here we describe how one can create a global forecast by simultaneously training a reservoir with an initial condition mapping. The result is a reservoir that can be used to predict the future dynamics starting at an arbitrary initial condition of the system. The creating of the initial condition mapping is made possible by breaking the input-signal in many smaller input signals, which we refer to as the windowed method.**

## I. INTRODUCTION

A reservoir computer is a machine learning model that can be used to predict the future state of time-dependent processes, in particular those associated with a dynamical system. Applications of reservoir computers include the modeling of dynamical systems, pattern classification, modeling robotics controllers, and event predictions. Recent applications include attractor reconstruction[1], predicting critical transitions, bifurcations, and phase in dynamical systems[2,3], separating chaotic signals[4,5], reconstructing dynamical states from partial observations[6,7], prediction and/or forecasting of chaotic systems[8,9].

The particular reservoirs we study are descendants of echo state networks[10] and liquid state machines[11], and are of the ODE variety considered by Lu[12] et al. To train this type of reservoir, data, in the form of an input-signal, is fed into the reservoir creating a response from each of the reservoir's internal nodes. These nodes, which are linked together to form the reservoir's internal network, respond both according to the input-signal and their interactions they have with other nodes. These responses are projected onto the original input-signal creating a trained version of the reservoir (see Section II).

The process of predicting the future state of a chaotic system is known as *attractor reconstruction*, which has been studied at least since the 1980s using a wide variety of machine learning models[13–17]. This includes neural networks and deep learning reservoir models. In these studies, reservoir computers were found to be competitive with deep learning methods, even outperforming RNNs and LSTMs on specific tasks including the reconstruction of dynamical systems[18–20]. In such tasks, reservoirs have been shown to be able to learn and reproduce characteristic features of chaotic systems including their first return maps, Lyapunov exponents, etc.[12,21–23].

In practice, a reservoir computer is trained on some input-signal and then used to predict the future state of this signal. We refer to this prediction as a *local forecast* as it predicts the dynamics of the input-signal only. The goal of this paper is to describe a process that creates a *global forecast* from an input-signal, one that can not only predict the future state of the original input-signal but can be used to learn a system in its entirety allowing one to predict the future state of that system starting at an arbitrary initial condition.

The notion of a global forecast as well as applications of this idea have been previously considered by a number of authors[24–26]. However, the difference between this method and the method presented here is that, in the former, predictions from an arbitrary initial point use a local section of the associated trajectory to create a forecast. The method proposed here uses no such local information and does not require any further processing once training is complete.

The idea is to create an initial condition mapping $\Phi : A \to R$ that relates the initial conditions in the state space $A$ of the dy-

[a)]Electronic mail: deinonychus@zookee.org
[b)]Electronic mail: quinlan.leishman@math.byu.edu
[c)]Electronic mail: wanderson@mathematics.byu.edu
[d)]Electronic mail: djpassey@unc.edu
[e)]Electronic mail: tpool@andrew.cmu.edu
[f)]Electronic mail: bwebb@math.byu.edu

namical system to points in the reservoir space $R$ in a way that respects the dynamics of the original system (see Section III). This initial condition mapping $\Phi$ is trained alongside the reservoir so that it is able to invert the reservoir's output mapping. To ensure that $\Phi$ is able to match the system's initial conditions with the reservoir's initial conditions we break the input-signal into smaller, potentially overlapping training signals which we refer to as *windows* (see Section IV). When the reservoir and initial condition mapping are trained over these windows they effectively learn how to make a local prediction near each of these windows. The result is a trained reservoir and a mapping $\Phi$ that can be used to make a global forecast of a system based on a single input-signal.

We refer to this process of training a reservoir and initial condition mapping on a set of windowed input-signals as the *windowed method*. To test the effectiveness of this method we investigate how the windowed method performs on the Lorenz, Rossler, and Thomas systems, specifically on recreating the global dynamics on each of their respective attractors. On these systems we compare the windowed method to the standard and partial methods. The *standard method* uses the standard reservoir computer with no initial condition mapping and no windows. The *partial method* uses an initial condition mapping but no windows.

We find that the windowed method significantly out performs both the standard and partial methods at creating a global forecast on each of the Lorenz, Rossler, and Thomas attractors (see Section VI). As the standard method has no effective way to relate system states to reservoir states this method fails at nearly every point to predict the dynamics of each system. The partial method has slightly better prediction accuracy, at least on subsets of the system's attractor, but is far less accurate overall when compared to the windowed method (cf. Figures 8, 9, and 11).

As our goal is to understand the prediction accuracy of the windowed method we also prove a few results related to error-bounds describing the window method's ability to accurately create a global forecast. These results give bounds based on Lipschitz and Lyapunov properties of the systems (see Theorems 1 and 2, respectively). These bounds help us to understand why we should expect the windowed method to work and also give sufficient conditions for improving the accuracy of a reservoir computer in terms of several simpler considerations.

The paper is organized as follows. In Section II we describe the specific reservoir computing model we consider and describe how this model is typically used to make local forecasts. In Section III we introduce the concept of a global forecast, one that begins at an arbitrary point in the system. This section describes how initial condition mappings are trained alongside the reservoir. In Section IV we describe the windowed method, which we use to create global forecasts. In Section V we describe our numerical setup and in Section VI we evaluate the window method's accuracy compare with the standard and partial methods. In Section VII we prove a number of error-bounds related to the accuracy of the windowed method, which we prove in the Appendix. In Section VIII we give some concluding remarks and describe a few open questions.

## II. RESERVOIR COMPUTING

A reservoir computer is a machine learning model used to predict the future state of time-dependent processes[27]. For an input-signal $\mathbf{u}(t) \in \mathbb{R}^m$ where $t \in [0,T]$ the standard goal is to use a reservoir to predict the future values of this time-series, i.e. predict $\mathbf{u}(t)$ for $t > T$. This is done in a sequence of three steps which we refer to as *processing*, *aggregation*, and *prediction*.

In the processing step, the nodes of a network are driven by the input-signal $\mathbf{u}(t)$ over the *training interval* $t \in [0,T]$. In reservoir computing, this network, which we refer to the as the reservoir's *processing network*, can be any network. Here, the network is given by a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ where the entry $\mathbf{A}_{ij} \in \mathbb{R}$ represents the weighted connection from node $j$ to node $i$.

In the model, the processing network's nodes evolve according to the differential equation

$$\frac{d}{dt}\mathbf{r}(t) = \gamma[-\mathbf{r}(t) + f\big(\rho\mathbf{A}\mathbf{r}(t) + \sigma\mathbf{W}_{\text{in}}\mathbf{u}(t)\big)] \qquad (1)$$

for $t \in [0,T]$ where $\mathbf{r}(t) \in \mathbb{R}^n$ represents the state of the nodes within the reservoir at time $t \in [0,T]$. The matrix $\mathbf{W}_{\text{in}} \in \mathbb{R}^{n\times m}$ in Equation (1) is fixed and sends a linear combination of the $m$-dimensional training data $\mathbf{u}(t) \in \mathbb{R}^m$ to each of the $n$ reservoir nodes. Similar to other reservoirs studied in previous work, we choose each entry of $\mathbf{W}_{in}$ uniformly according to the distribution $[\mathbf{W}_{\text{in}}]_{ij} \sim U(-1,1)$. The function $f : \mathbb{R}^n \to \mathbb{R}^n$ in Equation (1) is applied element-wise and $\gamma$, $\rho$, and $\sigma$ are non-negative scalar parameters of the reservoir.

As the input-signal $\mathbf{u}(t)$ appears in Equation (1), the nodes of the reservoir's processing network depend on, or are *driven* by, this time-series. The nodes' *response* to this input signal is given by the solution $\mathbf{r}(t)$ to this differential equation over the training period $t \in [0,T]$ (see Figure 1). In the training process, the initial reservoir state $\mathbf{r}(0) = \mathbf{r}_0$ is typically initialized randomly, with each coordinate drawn from a uniform distribution. The range of this distribution depends on the activation function $f$, since the latter influences the usual range of the reservoir computer nodes' states.

The second step in creating a trained reservoir is aggregating these network responses. The type of aggregation we consider is done by first discretizing the time interval $[0,T]$ into equal time-steps of length $\tau$. The result is the set of discrete time-steps $\{t_i\}_{i=0}^{\ell}$ where $t_i = i\cdot\tau$ and $\ell\cdot\tau = T$. Using this time-discretization, we compute the matrix

$$\mathbf{W}_{\text{out}} = \underset{\mathbf{W}\in\mathbb{R}^{\text{m}\times\text{n}}}{\operatorname{argmin}} \left[\sum_{i=0}^{\ell} \|\mathbf{W}\mathbf{r}(t_i) - \mathbf{u}(t_i)\|_2^2 + \alpha\|\mathbf{W}\|_2^2\right] \qquad (2)$$

that minimizes the sum on the right. Here, the parameter $\alpha > 0$ specifies the amount of regularization in this minimization process which is used to prevent overfitting.

The result is the mapping $\mathbf{W}_{\text{out}} \in \mathbb{R}^{m\times n}$, which is used to aggregate the network responses into the vector $\hat{\mathbf{u}}(t) =$

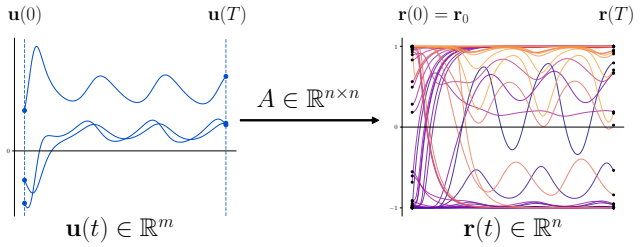


FIG. 1. Reservoir Processing: A reservoir computer is *trained* by first driving the nodes of its processing network, given by $A \in \mathbb{R}^{n\times n}$ (center), by an input-signal $\mathbf{u}(t) \in \mathbb{R}^m$ over the time interval $t \in [0,T]$ (left). The *response* $\mathbf{r}(t) \in \mathbb{R}^n$ (right) of the nodes to the input-signal over the same time interval is given by the solution to Equation (1).

$\mathbf{W}_{\text{out}}\mathbf{r}(t) \in \mathbb{R}^m$ (see Figure 2). In practice, $\alpha$ is typically small so that the *aggregated responses* can, roughly speaking, be considered to be a proxy for the input-signal, i.e. $\hat{\mathbf{u}}(t) \approx \mathbf{u}(t)$ over the training period $t \in [0,T]$.

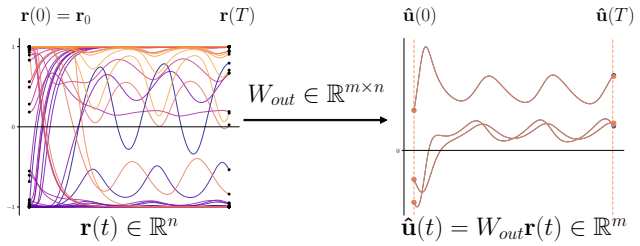


FIG. 2. Response Aggregation: The response vector $\mathbf{r}(t) \in \mathbb{R}^n$ (left) is aggregated by determining the matrix $\mathbf{W}_{\text{out}} \in \mathbb{R}^{m\times n}$ that minimizes the sum in Equation (2). The result is the aggregated response vector $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t) \in \mathbb{R}^m$ (right) that best approximates the input-signal $\hat{\mathbf{u}}(t)$ for $t \in [0,T]$.

The last step, the prediction step, is done by replacing the input-signal $\mathbf{u}(t)$ by the aggregate response vector $\hat{\mathbf{u}}(t)$ in Equation (1). This results in the autonomous differential equation

$$\frac{d}{dt}\hat{\mathbf{r}}(t) = \gamma[-\hat{\mathbf{r}}(t) + f\big([\rho\mathbf{A} + \sigma\mathbf{W}_{\text{in}}\mathbf{W}_{\text{out}}]\hat{\mathbf{r}}(t)\big)] \qquad (3)$$

which no longer depends on the training data. Because of this we refer to Equation (3) as the *trained reservoir*. This system has the new *aggregate network* given by the matrix $\hat{\mathbf{A}} = \rho\mathbf{A} + \sigma\mathbf{W}_{\text{in}}\mathbf{W}_{\text{out}} \in \mathbb{R}^{n\times n}$, as it combines the original adjacency matrix $\mathbf{A}$ with a scaled version of the matrix $\mathbf{W}_{\text{in}}\mathbf{W}_{\text{out}}$ that aggregates the responses $\hat{\mathbf{r}}(t)$.

Once a reservoir is trained, the goal is typically to make predictions regarding the future state of the input-signal beyond its training period. This is done by initializing the trained system at the value $\hat{\mathbf{r}}(T) = \mathbf{r}(T)$ at the end of the training period. This produces the time series $\hat{\mathbf{r}}(t)$ for $t \in (T,\infty)$ from which we create the *predicted time-series* $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t) \in \mathbb{R}^m$ for all future times (see Figure 3).

One can think of the predicted time-series $\hat{\mathbf{u}}(t)$ as a *forecast* of the future state of a given input-signal $\mathbf{u}(t)$. The forecast is

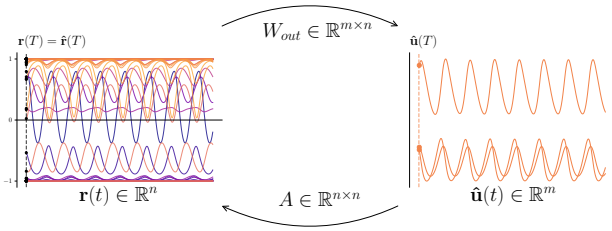


FIG. 3. Reservoir Prediction: Replacing the input-signal $\mathbf{u}(t)$ in Equation (1) by the aggregate response vector $\hat{\mathbf{u}}(t)$ results in the trained reservoir given by Equation (3). Once the input-signal is removed the trained reservoir drives itself giving the output $\hat{\mathbf{r}}(t) \in \mathbb{R}^m$ (right) and the predicted time-series $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t)$ that can be compared to the true time-series.

local in the sense that it is a continuation of the input-signal only. We call this type of forecast a *local forecast*, in contrast with *global forecasts* which we define in Section III.

In this paper we consider local and global forecasts of the following three chaotic dynamical systems: the Lorenz, Rossler, and Thomas systems. These are given by the following equations: The *Lorenz system*

$$\begin{aligned} dx/dt &= 10(y-x) \\ dy/dt &= x(28-z)-y \\ dz/dt &= xy-8z/3; \end{aligned} \qquad (4)$$

the *Rossler system*

$$\begin{aligned} dx/dt &= -y-z \\ dy/dt &= x+0.2y \\ dz/dt &= 0.2+z(x-5.7); \end{aligned} \qquad (5)$$

and the *Thomas system*

$$\begin{aligned} dx/dt &= \sin(y)-0.1998x \\ dy/dt &= \sin(z)-0.1998y \\ dz/dt &= \sin(x)-0.1998z. \end{aligned} \qquad (6)$$

In each system, parameters are chosen to yield chaotic dynamics[28–30].

Forecasts on such systems are typically accurate for only a short amount of time owning to the chaotic nature of the system's dynamics on its attractor. To determine how well our predicted time-series approximates the true dynamics of these systems, a useful measure of accuracy is a reservoir's valid prediction time.

**Definition 1.** ***(Valid Prediction Time)*** *Let $d(\cdot,\cdot)$ be a metric and let $\varepsilon > 0$ be a tolerance. The* valid prediction time (VPT) *of a prediction time-series $\hat{\mathbf{u}}(t)$ beginning at time $t = T$ associated with the signal $\mathbf{u}(t)$ is the largest value $t_* = T_* - T \geq 0$ such that the error*

$$d(\mathbf{u}(t),\hat{\mathbf{u}}(t)) \leq \varepsilon$$

*for all $t \in [T,T_*]$.*

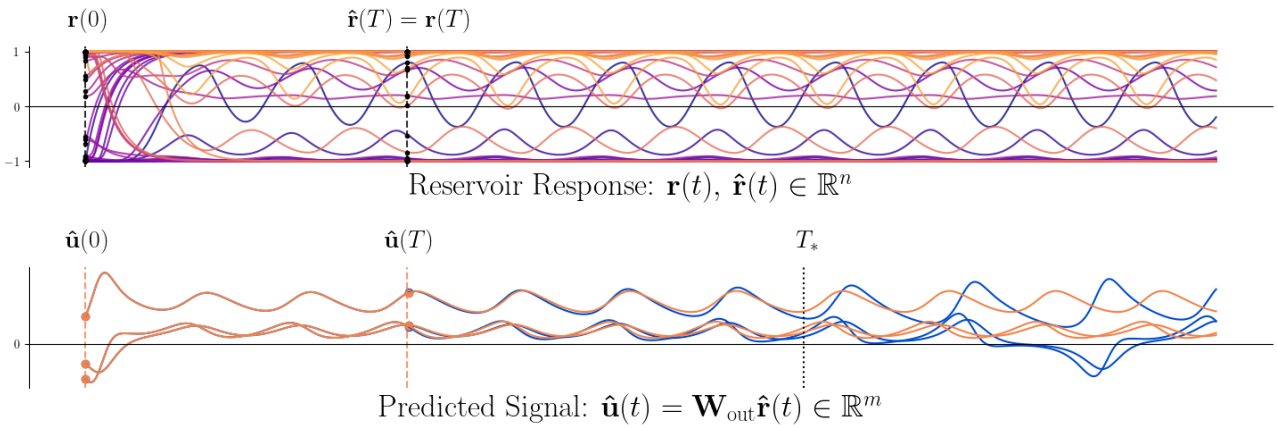


FIG. 4. Top: The response vectors $\mathbf{r}(t) \in \mathbb{R}^n$ for $t \in [0,T]$ and $\hat{\mathbf{r}}(t) \in \mathbb{R}^n$ for $t \geq T$ are shown for the reservoir in Example 1 trained on the Lorenz attractor. Bottom: A comparison between the true trajectory $\mathbf{u}(t)$ and the predicted trajectory $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t)$ for $t \geq T$ is shown in black and orange, respectively. The three curves indicate the $x$, $y$, and $z$-coordinates of each trajectory. The dashed line indicates the valid prediction time (VPT) $t_* = T_* - T = 2.45$, which is the first time at which the inequality $d(\hat{\mathbf{u}}(t), \mathbf{u}(t)) < \varepsilon$ is exceeded for the tolerance $\varepsilon = 0.5$ using the NRMSE-Norm $d(\cdot,\cdot)$ used in Section V (see Definition 3).

In other words, the VPT is a measure of how long the predicted time-series $\hat{\mathbf{u}}(t)$ stays close to the true time-series $\mathbf{u}(t)$ within the tolerance $\varepsilon$. This gives a measure of how well a trained reservoir can predict the future dynamics of the input signal beyond time $T$ when its training period ends.

**Example 1.** ***(Lorenz Reservoir)*** *Let $\mathbf{u}(t) \in \mathbb{R}^3$ for $t \in [0,T]$ be a finite trajectory sampled from the Lorenz attractor where $T = 2$. To train a computer reservoir on this input-signal we choose our processing network to be an Erdös-Renyi graph with $n = 50$ nodes with the probability $p = 0.1$ of an edge forming between any two nodes (see Section V for more details). For $\gamma = 8.780$, $\rho = 16.81$, and $\sigma = 0.187$ the resulting response vector $\mathbf{r}(t) \in \mathbb{R}^{50}$ for $t \in [0,T]$ is shown in Figure 4 (top). Computing $\mathbf{W}_{out}$ using $\alpha = 2.7 \times 10^{-8}$ yields the aggregate network with adjacency matrix $\hat{\mathbf{A}}$. From this we compute the aggregate response vector $\hat{\mathbf{u}}(t)$ shown together with the true trajectory $\mathbf{u}(t)$ for $t \geq T$ in Figure 4 (bottom).*

*The agreement between the actual and predicted trajectories in this example is reasonably good, at least initially. In particular, with $\varepsilon = 0.5$ the inequality $d(\hat{\mathbf{u}}(t), \mathbf{u}(t)) \leq \varepsilon$ holds only so long as $t \in [2, 4.45]$ meaning that, for the NRMSE-Norm $d(\cdot,\cdot)$ given in Section V (see Definition 3) and the tolerance $\varepsilon = 0.5$, we have a valid prediction time of length $t_* = 2.45$.*

In order to understand why (or why not) a reservoir computer makes accurate predictions, it is helpful to have a knowledge of some simpler, more concrete properties that lead to accurate predictions. For brevity, suppose we can write the dynamical system ODE as

$$d\mathbf{u}(t)/dt = g(\mathbf{u}(t)) \tag{7}$$

and the trained reservoir ODE as

$$d\hat{\mathbf{r}}(t)/dt = h(\hat{\mathbf{r}}(t)). \tag{8}$$

In this setting, both the true trajectory $\mathbf{u}(t)$ and prediction $\hat{\mathbf{u}}(t)$ can be thought of as solutions to differential equation initial value problems. So, a natural heuristic for $\mathbf{u}(t)$ and $\hat{\mathbf{u}}(t)$ to be near each other is for (1) their initial conditions to be nearby, and (2) their derivatives to be similar. In terms of the reservoir's initial condition $\hat{\mathbf{r}}_T = \hat{\mathbf{r}}(T)$, the first of these can be written as

$$\hat{\mathbf{u}}(T) = \mathbf{W}_{\text{out}}\hat{\mathbf{r}}_T \approx \mathbf{u}(T). \tag{9}$$

In terms of the ODE functions, the second condition can be written as

$$\hat{\mathbf{u}}'(t) = \mathbf{W}_{\text{out}}h(\hat{\mathbf{r}}(t)) \approx g(\mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t)) \tag{10}$$

for $t \geq T$; in words, the derivative of the prediction is roughly what it should be if it were an actual trajectory of the dynamical system.

In Section VII, we demonstrate that conditions (9) and (10) are in fact sufficient for accurate predictions, and we give several bounds on the prediction error in terms of these. Consequentially, these two conditions will be helpful for understanding why certain training methods lead to better or worse accuracy as we proceed through this paper.

As reservoir computers are known to have the ability to create accurate local forecasts, it is perhaps unsurprising that for such reservoirs, both conditions (9) and (10) typically hold. However, it may be surprising that using only a single training signal, we can predict the future state of the system starting at an arbitrary initial condition, even one that is not a part of the input signal. This global forecasting method is described in the following section.

## III. GLOBAL FORECASTS

Suppose we have a reservoir trained on an input-signal $\mathbf{u}(t)$ over the time interval $t \in [0,T]$. A local forecast $\hat{\mathbf{u}}(t)$ starts at

the point $\hat{\mathbf{u}}(T) \approx \mathbf{u}(T) \in \mathbb{R}^m$ and thereafter is a prediction of the input-signal (cf. Example 1). In this section, we discuss whether is it possible to create a reservoir computer using the same input-signal to predict the future dynamics of the system starting at an arbitrary point $\mathbf{v} \in \mathbb{R}^m$. We refer to this as the *global forecasting task*.

In this and following sections, we will use $\mathbf{u}(t)$ for $t \in [0,T]$ to denote the input or training signal, with initial condition $\mathbf{u}(0) = \mathbf{u}$. We let $\mathbf{v}(t)$ for $t \geq T$ denote the signal starting at $\mathbf{v}(T) = \mathbf{v}$ that we are attempting to predict; and $\hat{\mathbf{v}}(t)$ to denote our prediction for this initial condition.

To create a global forecast, we first need a way to relate any initial condition $\mathbf{v} \in \mathbb{R}^m$ to a point $\hat{\mathbf{r}}(T) \in \mathbb{R}^n$ in *reservoir space*. It is very unlikely that selecting the initial condition at random will work. Rather, we require a map $\Phi : A \to R$ that will map initial conditions $\mathbf{v} \in A \subset \mathbb{R}^m$ on the attractor $A$ of the dynamical system to initial conditions $\hat{\mathbf{r}}_T \in R \subset \mathbb{R}^n$ of the reservoir space $R$. We call such a function $\Phi$ an *initial condition mapping*.

To create such a map, note that conditions (9) and (10) are still applicable to the prediction accuracy for the global forecasting task, with $\mathbf{u}$ replaced by $\mathbf{v}$ as needed. We can write these conditions in terms of $\Phi$ as follows.

First, we need the reservoir's prediction to start at the the correct initial condition. We have $\hat{\mathbf{r}}_T = \Phi(\mathbf{v})$, so applying condition (9) we require that

$$\mathbf{W}_{\text{out}}\Phi(\mathbf{v}) \approx \mathbf{v}. \tag{11}$$

Equivalently, we require $\mathbf{W}_{\text{out}}$ and $\Phi$ to be approximate pseudoinverses. Additionally, for (11) to hold for any $\mathbf{v} \in A$ on the attractor, we also need for $\Phi$ to be injective. Since $R \subset \mathbb{R}^n$ has, in practice, a higher dimension than $A \subset \mathbb{R}^m$, then for the functions we consider this is easy to guarantee.

Second, we need the reservoir's dynamics to be accurate at the very least along the trajectory of the reservoir states starting from the chosen initial condition $\Phi(\mathbf{v})$. This condition can be written in terms of $\Phi$ and $\mathbf{W}_{\text{out}}$ as

$$\mathbf{W}_{\text{out}}h(\hat{\mathbf{r}}(t)) \approx g(\mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t)) \tag{12}$$

for $t \geq T$ and for $\hat{\mathbf{r}}_T = \Phi(\mathbf{v})$.

An additional necessary condition for Equation (12) to hold is the following. It is easily verified that if the range of each coordinate of the activation function $f : \mathbb{R}^n \to \mathbb{R}^n$ is contained in the interval $[a,b]$, then each coordinate of the reservoir state will be attracted towards this interval and cannot leave it once entered (cf. Equation 1). We note this is the case for most of the standard activation functions (e.g. sigmoid, hyperbolic tangent, etc.). In other words, all states outside $[a,b]^n \subset \mathbb{R}^n$ will initially have transient dynamics and all of the non-transient reservoir dynamics will be inside this set. These transient states are generally unhelpful for learning the dynamics of the system's attractor as their short-term behavior away from the attractor is very different from the long-term behavior on or near the attractor.

Thus, any initial condition mapping $\Phi$ with property (12) will need to map the dynamical system attractor into $[a,b]^n$. If this is not the case, then for points $\mathbf{v} \in A$ such that $\Phi(\mathbf{v}) \notin [a,b]^n$, the short-term behavior of the reservoir will likely not approximate the behavior of the dynamical system very well. This restricts the class of functions that are likely to work well as initial condition mappings.

In Section VII we show that these two conditions, Equations (9) and (10), are sufficient for accurate global predictions. This suggests that conditions (11) and (12), as well as requiring injectivity and that the image is contained in $[a,b]^n$, are useful criteria for an initial condition mapping.

An effective way to make an initial condition mapping $\Phi : A \to R$ satisfying conditions (11) and (12) is to simply fix the mapping beforehand and use that same mapping in the training process. Then, when $\mathbf{W}_{\text{out}}$ is created, we can make it map the training reservoir initial condition $\mathbf{r}_0 = \Phi(\mathbf{u})$ back to the input signal initial condition, as well as map reservoir dynamics to system dynamics as before. The result of this process is that conditions (11) and (12) are satisfied, at least for points $\mathbf{v} \in A$ near $\mathbf{u}$.

Rather than thinking of $\Phi$ as finding points in the reservoir space that have the correct dynamics, we can think about this approach as forcing $\mathbf{W}_{\text{out}}$ to learn the inverse of $\Phi$, as well as mapping the reservoir dynamics to the dynamical system's dynamics, which respectively fulfills conditions (11) and (12). The desired inverse relationship between $\Phi$ and $\mathbf{W}_{\text{out}}$ is illustrated in Figure 5.

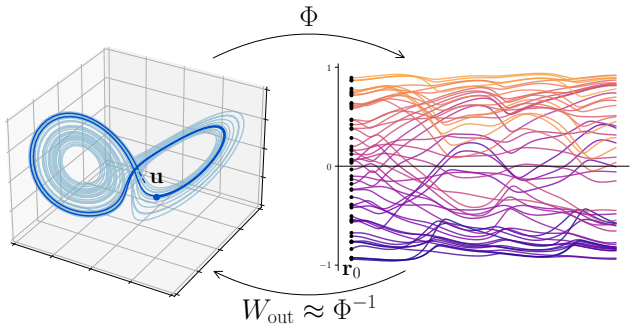


FIG. 5. An illustration of the relationship between the initial condition mapping $\Phi : \mathbb{R}^3 \to \mathbb{R}^n$ and the output mapping $\mathbf{W}_{\text{out}} : \mathbb{R}^n \to \mathbb{R}^3$ for the Lorenz attractor. On the left is the training signal with its initial condition $\mathbf{u}$ marked. The initial reservoir condition $\mathbf{r}_0 \in \mathbb{R}^n$ is shown on the right along with the reservoir's response to this initial condition. By fixing $\Phi$ before the training process, $\mathbf{W}_{\text{out}}$ will approximate the inverse of $\Phi$, meaning that it maps an arbitrary reservoir initial conditions $\hat{\mathbf{r}}_T = \Phi(\mathbf{v})$ back to the corresponding original point $\mathbf{v} \in \mathbb{R}^3$ on the dynamical system. This property enables the finding of a reservoir state that satisfies (11) and (12) given any initial condition on the attractor and allowing the reservoir computer to accurately predict the dynamics from any such point.

Motivated by the ideas discussed above, we have found several maps $\Phi$ that work well in practice. We will, for the sake of simplicity, restrict ourselves to considering the following map:

$$\Phi(\mathbf{v}) = f(\sigma W_{\text{in}}\mathbf{v}) \tag{13}$$

where $f$ is the activation function of the reservoir ODE. This mapping can be seen as sending an arbitrary $\mathbf{v} \in A$ to an ap-

proximation of a fixed point of the untrained reservoir ODE with constant input-signal $\mathbf{u}(t) = \mathbf{v}$ (cf. Equation (1)). More precisely, this is the fixed point if the term $\rho A\hat{\mathbf{r}}(t)$ in Equation (1) is set to zero.

It is very easy to verify that this map $\Phi$ is injective. It will be so as long as $f$ is injective and $\mathbf{W}_{\text{in}}$ has full rank, which is the case for almost all matrices of its dimensions. This map $\Phi$ also has the property that it maps the attractor $A$ into the reservoir's attracting region $[a,b]^n$. It is thus a reasonable choice based on the discussion above.

With the addition of $\Phi$ to the usual reservoir computer algorithm, we are closer to our goal of creating a global forecast (see part (b) of Figure 9, where this approach is called the partial training method; this is in contrast to the windowed training method, discussed in the next section). As can be seen, with this training method the reservoir computer usually is able to make accurate predictions on a small region of the attractor, usually near the initial condition $\mathbf{u}(0)$ of the original training signal. We will return to this pattern in the next section.

In contrast, it turns out to be very difficult to choose a map $\Phi$ that satisfies both of these conditions after the reservoir has been trained. Condition (11) regarding initial conditions is not hard to satisfy by itself. For example, one can just choose $\Phi = \mathbf{W}_{\text{out}}^{\dagger}$, where $\mathbf{W}_{\text{out}}^{\dagger}$ is any pseudoinverse of $\mathbf{W}_{\text{out}}$. Condition (12), on the other hand, is where the difficulty lies. A generic pseudoinverse will not map $A$ into $[a,b]^n$; even if carefully chosen to do so, and there is no reason to expect that condition (12) will be satisfied. For this reason, given a reservoir computer that has already been trained, there does not seem to be a good approach to choosing a map $\Phi$ with this property. By choosing the function $\Phi$ beforehand, we instead force our minimization process (2) to create $\mathbf{W}_{\text{out}}$ with the required properties.

## IV. WINDOWED TRAINING METHOD

We now consider how to increase the number of initial conditions on the dynamical system attractor $A$ at which the reservoir computer can make accurate predictions. When using an initial condition mapping, the major issue we face is that $\mathbf{W}_{\text{out}}\Phi(\mathbf{v}) \approx \mathbf{v}$ only on a small region of the attractor $A$. This happens because there is only a single reservoir initial condition $\mathbf{r}_0 = \Phi(\mathbf{u})$ that $\mathbf{W}_{\text{out}}$ needs to map back to $\mathbf{u}$. The regression used to find $\mathbf{W}_{\text{out}}$ results in $\mathbf{W}_{\text{out}}\Phi(\mathbf{u}) \approx \mathbf{u}$. However, this is all that is guaranteed. The matrix $\mathbf{W}_{\text{out}}$ will approximate $\Phi^{-1}$ only near $\mathbf{r}_0$, so condition (11) will only hold for other initial conditions $\mathbf{v} \in A$ near the original $\mathbf{u}$.

The solution to this problem is to add more pairs of points that $\mathbf{W}_{\text{out}}$ needs to map correctly from the reservoir space to the attractor. As it turns out, a useful way of doing this while also allowing $\mathbf{W}_{\text{out}}$ to learn the system's dynamics correctly is to break our training signal into *windows*.

**Definition 2.** ***(Training Windows)*** *A set of* windows *for a training signal $\mathbf{u}(t)$ defined for $t \in [0,T]$ is a finite collection of intervals $\{\Psi_i\}_{i=1}^{\ell}$ that cover $[0,T]$. That is,*

$$\bigcup_{i=1}^{\ell} \Psi_i = [0,T].$$

*We denote each interval as $\Psi_i = [a_i, b_i]$ and note that the windows are not required to be disjoint.*

We parameterize the windows by two variables: *window size* $L \in (0,T]$ and *overlap* $v \in [0,1)$. Window size is the length of each window, i.e. $L = b_i - a_i$ for all $i = 1,\ldots,\ell$. The overlap $v$ is the proportion of how much each window overlaps with the previous one. Thus, each window is given inductively by $[a_1,b_1] = [0,L]$ and $[a_{i+1},b_{i+1}] = [a_i + (1-v)L, a_{i+1} + L]$ for $i = 1,\ldots,\ell-1$, where there are $\ell = 1 + \left\lfloor \frac{T-L}{(1-v)L} \right\rfloor$ windows in total.

To create a global forecast, we proceed similarly to the method described in Section II of sequentially processing, aggregating, and predicting to produce the trained reservoir and the initial condition mapping:

- **Processing:** The processing step is done separately on each window. For each window $\Psi_i = [a_i, b_i]$, we choose the reservoir initial condition as $\Phi(\mathbf{u}(a_i))$, run the reservoir until time $b_i$, and collect all of the responses.
- **Aggregation:** Aggregation is done on all windows together at once. Since we are using least squares for this step, where it does not matter whether adjacent data points are related to each other, this poses no difficulties. We denote the time discretization of window $\Psi_i$ as $\{t_{ij}\}_{j=1}^{\ell_i}$ and the reservoir response on $\Psi_i$ as $\mathbf{r}_i$. Then the new least squares problem for determining $\mathbf{W}_{\text{out}}$ can be written as follows:

$$\begin{aligned}\mathbf{W}_{\text{out}} &= \underset{\mathbf{W}\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \left[ \sum_{i=1}^{\ell} \sum_{j=0}^{\ell_i} \left\| \mathbf{W}\mathbf{r}_i(t_{ij}) - \mathbf{u}(t_{ij}) \right\|_2^2 + \alpha \|\mathbf{W}\|_2^2 \right] \\ &= \underset{\mathbf{W}\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \left[ \sum_{i=1}^{\ell} \left( \|\mathbf{W}\Phi(\mathbf{u}(t_{i,0})) - \mathbf{u}(t_{i,0})\| \right.\right. \\ &\qquad \left.\left. + \sum_{j=1}^{\ell_i} \left\| \mathbf{W}\mathbf{r}_i(t_{ij}) - \mathbf{u}(t_{ij}) \right\|_2^2 \right) + \alpha \|\mathbf{W}\|_2^2 \right]. \end{aligned} \tag{14}$$

  On each window, the reservoir response is mapped back onto the training signal of the same window.
- **Prediction:** The prediction step is done as before, with $\hat{\mathbf{r}}_T = \Phi(\mathbf{v})$ (see Equation (3)).

We call this algorithm the *windowed training method*. An illustration of this training process is given in Figure 6.

The benefit of using the windowed training algorithm is that at the start of each window at time $t = a_i$, a reservoir initial condition is chosen as $\Phi(\mathbf{u}(a_i))$. For each of these initial conditions, the minimization process for $\mathbf{W}_{\text{out}}$ requires that it maps each of these back to $\mathbf{u}(a_i)$. Thus, $\mathbf{W}_{\text{out}}\Phi(\mathbf{u}(a_i)) \approx \mathbf{u}(a_i)$ for each $1 \le i \le \ell$. As a result, $\mathbf{W}_{\text{out}}$ will approximate the pseudoinverse of $\Phi$ for points in $A$ near any of the $\mathbf{u}(a_i)$,

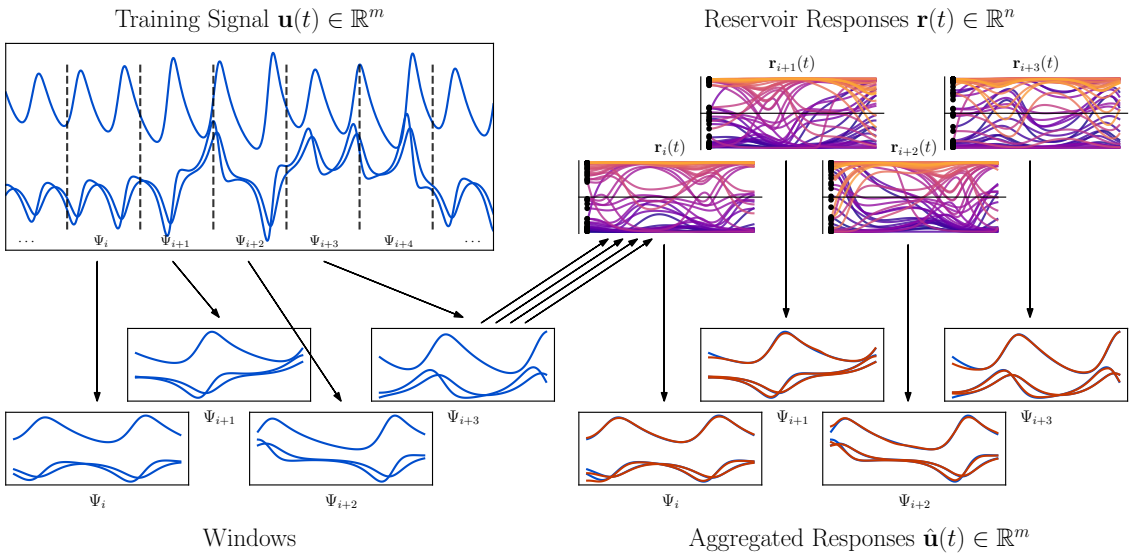


FIG. 6. An illustration of the windowed training method using windows $\{\Psi_i\}_{i=1}^{\ell}$. The input-signal $\mathbf{u}(t) \in \mathbb{R}^m$ (top left) is first broken apart into possibly overlapping windows (bottom left). For each of the windows $\Psi_i = [a_i, b_i]$, a reservoir initial condition is chosen using $\Phi : \mathbb{R}^m \to \mathbb{R}^n$, and the reservoir's response to the input-signal $\mathbf{u}(t)$ is found (top right). Finally, the responses across all of the window are aggregated back using Tikhanov regression to approximate the individual training signals (bottom right), obtaining the output mapping $\mathbf{W}_{\text{out}}$ (see Equation (14)). For each window, the reservoir response is aggregated back to the training signal on just that window, regardless of whether that window overlaps with other windows. Because the reservoir computer is trained using many initial conditions spread across the attractor then $\mathbf{W}_{\text{out}}$ should be an accurate approximation of $\Phi^{-1}$. This allows the reservoir computer to make much more accurate predictions across the attractor when compared to the standard method (cf. Figures 8 and 9).

which is the mechanism that allows us to create a global forecast from a single training signal. This can be thought of as allowing the reservoir computer to learn how to make local predictions in the vicinity of each of the windows.

In Figure 7 we give an example of the effect of using this training algorithm. The top of the figure illustrates the training $\mathbf{u}(t)$ signal and the points used as the start of each window. In the bottom of the figure we see that in the regions with more windows global predictions are more accurate than those where there are less windows. This is due not only to the dynamics being more accurately learned near the training signal, but also to the initial conditions being mapped more accurately.

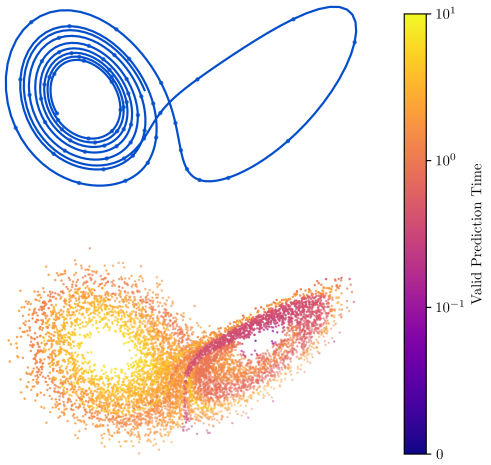


FIG. 7. Top: A training signal taken from the Lorenz attractor is shown which is used to train a reservoir computer via the windowed method. The dots along the path indicate points at the start of each window. Bottom: A total of 8,000 initial conditions are sampled from the Lorenz attractor. The reservoir computer trained on the signal shown above is used to predict the trajectory starting from each point and each point is colored according to the resevoir's VPT starting at each point. Note that this training signal has comparatively few windows on the right half of the attractor, which correlates with the lower valid prediction times for most of the initial conditions on that side.

## V. METHODOLOGY AND NUMERICAL SETUP

We now describe our numerical experiments to compare the following three training algorithms:

1. **Standard Method:** The *standard method* is the original method described in Section II typically used to create local forecasts. It uses no initial condition mapping or windows during training, and the pseudoinverse of $W_{\text{out}}$ for choosing initial conditions for the global forecasting task.

2. **Partial Method:** The *partial method* creates a global forecast using a fixed initial condition map $\Phi$ but no windows.

3. **Windowed Method:** The *windowed method*, creates a global forecast using both and initial condition mapping $\Phi$ and a set of windows $\{\Psi_i\}_{i=1}^{\ell}$.

We include the partial method to better isolate the effect of the initial condition mappings from the full window method. We compare these three methods on three chaotic systems: the Lorenz attractor, the Rossler system, and the Thomas system given in Section II (see Equations (4)–(6)).

Our primary goal is to compare the ability of these three algorithms to predict a time series $\mathbf{v}(t)$ where this is the solution to the original system starting at $\mathbf{v} \in A$ at time $t = T$, i.e. the *true* trajectory starting at $\mathbf{v}$. We let $\hat{\mathbf{v}}(t) = \mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t)$ denote our *predicted* trajectory starting at $\mathbf{v}$, where $\hat{\mathbf{r}}(t)$ is the response of the reservoir. As before, we denote the input-signal used to train the reservoir as $\mathbf{u}(t)$ for $t \in [0,T]$.

Our goal is to understand how the error $\|\hat{\mathbf{v}}(t) - \mathbf{v}(t)\|$ grows for a wide range points $\mathbf{v} \in A$. Specifically, we want to understand how the VPTs for the global prediction task are distributed for the Lorenz, Rossler, and Thomas systems.

To examine these distributions we must first pick a metric $d(\cdot,\cdot)$ and a tolerance $\varepsilon > 0$ (see Definition 1). For the tolerance we choose $\varepsilon = 0.5$. For the metric we choose the normalized root mean square error (NRMSE).

**Definition 3.** ***(NRMSE-Norm)*** *For a trajectory $\mathbf{u}(t)$ and an approximation $\hat{\mathbf{u}}(t)$ for $t \geq T$, let $\sigma_i$ be the sample standard deviation of the ith coordinate of $\mathbf{u}(t)$. Let $S = \text{diag}(\sigma_1, \ldots, \sigma_m)$ be the diagonal matrix of standard deviations. Then the NRMSE at time $t$ is*

$$\text{N}(\mathbf{u}(t), \hat{\mathbf{u}}(t)) = \frac{1}{\sqrt{m}} \left\| S^{-1}(\mathbf{u}(t) - \hat{\mathbf{u}}(t)) \right\|_2.$$

This particular version of NRMSE normalizes each coordinate of the true trajectory separately across time, and then computes the RMSE across the coordinate axes.

Before we can conduct any experiments we also need to choose hyperparameter values. The three types of hyperparamters we need are network hyperparameters $p_{net}$, reservoir parameters $p_{res}$, and the windowed parameters $p_{win}$. Here network hyperparameters describe the processing networks we will use in our numerical experiments. The type of processing network we use are random digraphs or *directed Erdös-Renyi graphs* commonly used in reservoir computing[12]. The random digraph model is a model in which $n \in \mathbb{N}$ is the number of network nodes and $p \in [0,1]$ is the probability that a *directed* edge is placed from node $i$ to node $j$ in the network for any pair of possibly non-distinct nodes $i$ and $j$. The network parameters we consider are the pair $p_{net} = (n,c)$ where $c = p \cdot n$ is the mean in/out-degree of the network.

For our experiments we selected nine combinations that parameterize the network adjacency matrix $A$ given by

$$\begin{aligned} n &\in \{500, 1000, 2000\} \\ c &\in \{0.1, 1.0, 2.0\}. \end{aligned}$$

As mentioned in Section II the observation matrix $\mathbf{W}_{\text{in}}$ for each reservoir computer is selected randomly with each entry uniformly drawn from $[-1,1]$. The activation function $f$ of the network we choose to be $f(x) = \text{sigm}(x) = (1+e^{-x})^{-1}$, the sigmoid function, applied component-wise. For the partial and windowed training methods, we choose $\Phi$ as described in Equation (13).

The remaining hyperparameters of the model are found using Bayesian hyperparameter optimization (BHO). We use this method as it been demonstrated to be much more efficient in higher dimensional optimization than more brute force approaches such as grid search methods[31]. The parameters we find using this method are the reservoir parameters $p_{res} = (\alpha, \gamma, \rho, \sigma)$ and the windowed parameters $p_{win} = (L, v)$. Here the windowed parameters are only found when using the windowed method.

This optimization is run twice for each parameter combination of $p_{net}$: once to find parameters that maximize the VPT for making local predictions, and once to find parameters that maximize the average VPT for making a global forecasts. For the Bayesian hyperparameter optimization, we sampled 100 hyperparameter configurations using this method. For each configuration, we trained $2^8$ reservoir computers using these parameters and had each one make a prediction of the chosen type, using either the standard, partial, or windowed training method. Unless noted otherwise, each of the experiments below used the optimized hyperparameters found in this way.

| Local | Standard | Partial | Windowed |
|---|---|---|---|
| Lorenz | 3.27 | 14.82 | 0.63 |
| Rossler | 25.00 | 9.29 | 18.48 |
| Thomas | 22.45 | 5.19 | 20.29 |
| Global | Standard | Partial | Windowed |
| Lorenz | 22.16 | 0.10 | 2.18 |
| Rossler | 12.13 | 22.70 | 18.4 |
| Thomas | 6.50 | 0.10 | 14.17 |

TABLE I. Optimized $\rho$ (spectral radius) values found by BHO for each method and attractor.

The optimized $\rho$ values found by Bayesian hyperparameter optimization are reported in Table I. For the windowed training method, the optimal window size and overlaps were $L = 0.45$ and $v = 0.80$ for the Lorenz system, $L = 22.29$ and $v = 0.95$ for the Rossler system, and $L = 4.07$ and $v = 0.10$ for the Thomas system.

All experiments were run in Python and are available on a Github repository.[32] The reservoir computer implementation we use is found in the `rescomp` Python package.[33] We used the LSODA algorithm from ODEPACK, wrapped by the Python package `scipy`, for numerical integration for the dynamical systems and for the reservoir ODE. Bayesian hyperparameter optimization was performed using the implementation in the `parameter-sherpa` Python package.[34]

## VI. RESULTS

In this section we provide the results of our numerical experiments, which compare the standard, partial, and windowed methods' ability to accurately forecast the dynamics on

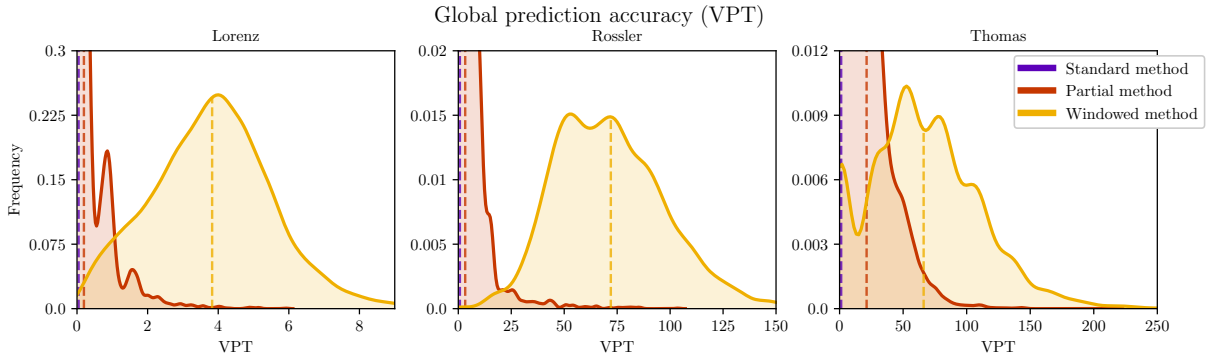


FIG. 8. The distributions of the global prediction accuracy (VPT) for different training methods for the Lorenz (left), Rossler (center), and Thomas (right) systems are shown. The dashed vertical lines indicate the mean VPT for each method. In each case the windowed method substantially outperforms the other two methods while the partial method moderately outperforms the standard method. The standard method effectively achieves a VPT of zero on this task. Here, the training signals from the Lorenz system had length $T = 6.6$, those from the Rossler system had length $T = 165$, and those from the Thomas system had length $T = 660$.

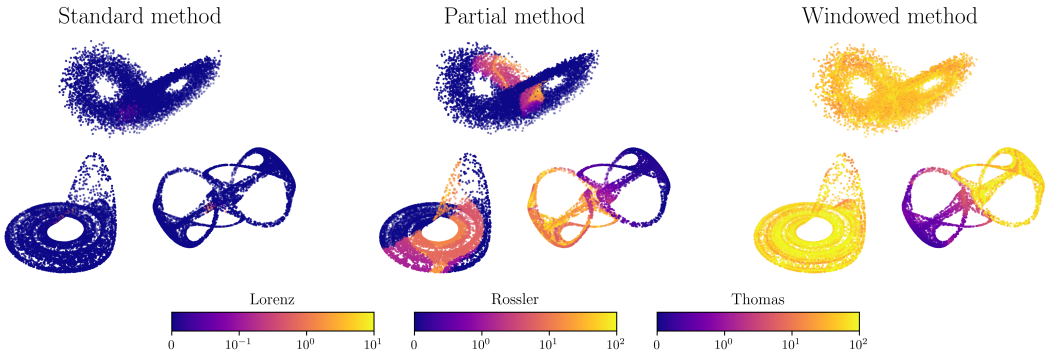


FIG. 9. For each of the Lorenz, Rossler, and Thomas systems, a reservoir is trained using each of the standard, partial, and windowed training methods, respectively. Then, we sampled $8 \cdot 10^3$ points from each attractor and had each point colored by its associated VPT as in Figure 7. On each attractor yellow indicates longer accuracy times, and purple shorter, with the color scales as given at the bottom of the figure for each system. The standard method appears to be unable to create accurate global predictions. The partial method performs somewhat better, obtaining moderate VPTs on parts of the attractor. The windowed method can be seen to predict very accurately and uniformly across the whole attractor for the Lorenz and Rossler systems, and half of the attractor for the Thomas system. For the latter, the difference is likely due to the fact that trajectories on the Thomas attractor rarely pass from one half of the attractor to the other. Hence, the reservoir computer is typically only trained on one half of the attractor and generally cannot accurately predict characteristics of the other half.

the Lorenz, Rossler, and Thomas attractors. Our experiments are broken into three parts; global predictions, the uniformity or nonuniformity of global predictions across an attractor, and local forecasts for comparison.

### A. Global Prediction Accuracy

Here we compare the standard, partial, and windowed algorithms on the global prediction task. The trained reservoir's initial condition $\hat{\mathbf{r}}_T$ for prediction is set equal to $W_{\text{out}}^{\dagger}\mathbf{v}$ for the standard method, and to $\Phi(\mathbf{v})$ for the partial and windowed training algorithms. For each training method, system, and choice of network parameters $p_{net}$, we train $8 \cdot 10^3$ reservoir computers using the optimized hyperparameters obtained through BHO. We then have each reservoir computer make a prediction from a randomly-selected point on the attractor. For each of these, we computed the valid prediction time of the prediction.

The distribution of valid prediction times resulting from this process for $n = 1000$, $c = 1.0$ are shown in Figure 8. The results for the other network parameter choices were similar.

The experimental results indicate that the windowed training method is much more able to perform the global prediction task when compared to the standard and partial methods for each system we consider. As can be expected for the vast majority of initial conditions, the standard training method's VPT is equal to zero. The partial method falls into a middle ground between the two, which is also as expected, where many of the predictions for this method still have very low VPT, although some are substantially higher. The reasons for this are explored further in the following section (cf. Figure

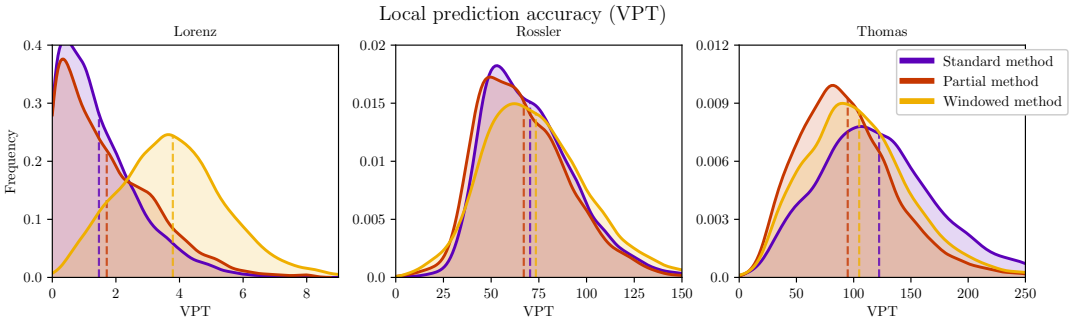


FIG. 10. The distribution of the local prediction accuracy (VPT) for different training methods for the Lorenz (left), Rossler (center), and Thomas (right) systems. The dashed vertical lines indicate the mean VPT for each method. Since this task does not require converting states on the attractor into reservoir states via some initial condition mapping $\Phi$, there is no reason the expect that either the partial or windowed method should outperform the standard method. On the Rossler and Thomas systems, all three methods perform similarly, as expected. However, in the Lorenz system the windowed method performs noticeably better than the standard and partial methods. This phenomena is further explored in Figure 11. Here, the training signals from the Lorenz system had length $T = 6.6$, those from the Rossler system had length $T = 165$, and those from the Thomas system had length $T = 660$.

9).

By comparing with Figure 10, where the analogous experiment is carried out for the local prediction task (described below), we see that the typical VPTs obtained using the windowed training method on the global prediciton task on the Lorenz and Rossler systems are comparable to or better than the typical VPTs obtained by the standard training method on the local prediction task. For the Thomas system, the typical VPTs of the windowed method for global forecasts are lower than those of the standard method for local forecasts. Nevertheless, the windowed method is still a substantial improvement on the standard method for the global prediction task.

### B. Uniformity across the Attractor

The second question we investigate is how accurate the windowed method is on different parts of the attractors we consider. That is, given a reservoir trained on a single input-signal, we explore how well the windowed method performs over the whole attractor compared to the other methods. To test this, for each of the three systems considered, we trained reservoir computers using each of the three training methods, with the hyperparameters equal to the optimal choice described above. We then randomly sample $8 \cdot 10^3$ points from the attractor of each dynamical system and computed the VPT associated with each.

In Figure 9, we visualize the results of a single realization of this process. In this figure, the initial starting point of each trajectory considered is plotted. Each point is colored according to the VPT of the reservoir computer's prediction. Yellow indicates more accurate predictions, while purple indicates inaccurate predictions.

Our numerics indicate that the windowed method is generally able to learn how to do global predictions across the whole attractor and that this accuracy is quite uniform for the typical input-signal of lengths $T = 6.6$, $T = 165$, and $T = 660$ we use for the Lorenz, Rossler, and Thomas attractors, respectively. These results are consistent over repeated experiments where new reservoirs are trained each time. We note that the Thomas system is somewhat of an exception to this, which is likely due to the nature of the system where trajectories rarely switch from one half of the attractor to the other. As a consequence, the training trajectories typically stay exclusively on one half of the attractor, so the reservoir is unable to accurately learn the other side of the attractor even with the windowed training method. This does point out the phenomena that reservoir computers (trained with or without the window method) do not learn regions of the dynamical system not visited by the training signal, which is not surprising.

Based on these experiments, the standard training method fails completely at the global task. The partial method, however, does show an interesting middle ground here. It has only certain regions that the reservoir computer can predict accurately, corresponding to the areas near the state of the system that was mapped to the original reservoir initial condition. In the remainder of the attractor, the partial method performs quite poorly.

Here, the difference between the plots for the partial method and the windowed method demonstrates the importance of mapping many initial conditions in order for the reservoir computer to be able to accurately learn the inverse mapping.

### C. Local Prediction Accuracy

In addition to testing for global prediction accuracy, we also are interested in how well reservoir computers trained with our proposed algorithm perform at the original reservoir task of making local predictions. The setup of these experiment is analogous to that of the global experiments described in Section VI, Subsection A. For each training method, system, and choice of network parameters, we train $8 \cdot 10^3$ reservoir computers using the optimized hyperparameters found via BHO.

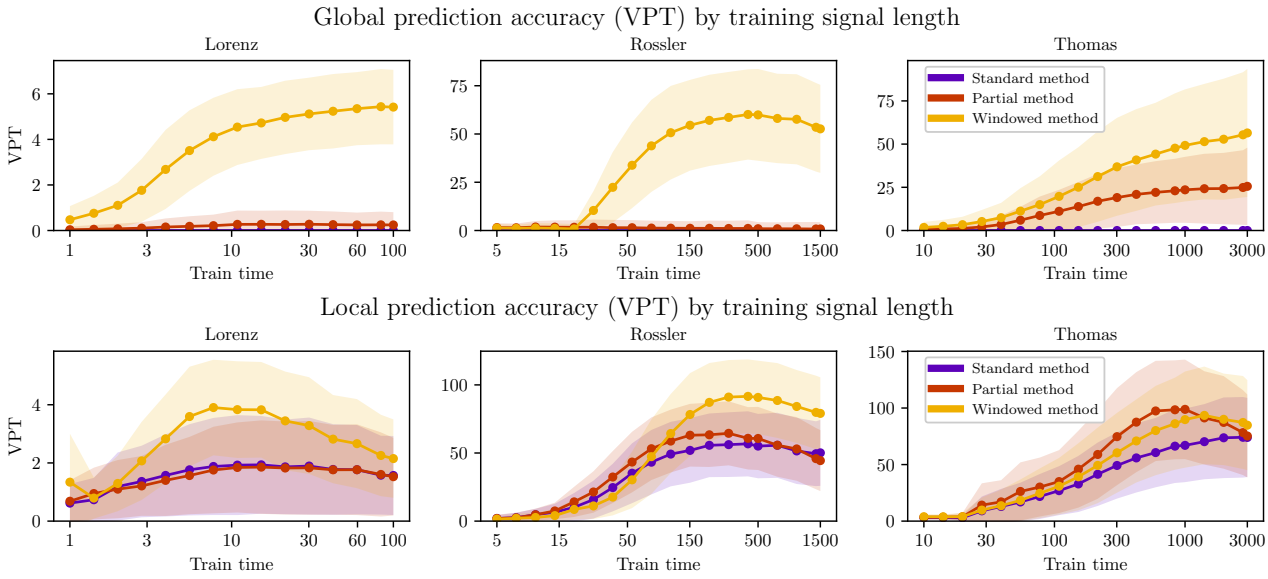


FIG. 11. Top: Comparison of the global prediction accuracy with optimized hyperparameters of the different training methods across varying training signal lengths. Note that the train time is plotted logarithmically. The windowed method is consistently better than the other methods for all training signal lengths other than very short times. Bottom: Comparison of the local prediction accuracy with optimized hyperparameters of the different training methods across varying training signal lengths. On the Lorenz system, the windowed method outperforms the other methods for the majority of training lengths, before decreasing to be similar for longer signal lengths. On the Rossler system, the three methods perform similarly for short training signal lengths, but the windowed method outperforms the other two for longer signal lengths. On the Thomas system, all three methods perform similarly, with the partial method slightly outperforming the others. The reason for these variations is an open question.

For each we predict the continuation of the training signal using the final reservoir state from training as the initial condition, and computed the VPT of each prediction.

The results of this process for $n = 1000$ and $c = 1.0$ are summarized in Figure 10. The results for the other network parameter choices are qualitatively similar.

Our initial hypothesis was that the windowed method would perform potentially worse than both the standard and partial methods due to trade-offs in over fitting. On the Rossler and Thomas systems all three methods perform with nearly the same average accuracy. For the Lorenz system the windowed method outperforms the others (see Figure 10). The reason for this is currently an open question but is possibly related to some type of underfitting or overfitting due to the specific training time, which we investigate in the following section (cf. Figure 11).

### D. Dependence on Training Signal Length

To understand how VPTs are effected by training times using our three different methods, we investigate how varying the time $T > 0$ changes the average VPT for both the local and global predictions over each attractor. The setup for this experiment is similar to those described in Sections VI A and VI C. For each method and for each system we choose the hyperparameters using BHO with $n = 1000$ and $c = 1.0$. We then select a range of training signal lengths for each system, evenly spaced logarithmically as shown in Figure 11. Then for each training signal length on each system and for each training algorithm we trained $2^{12}$ reservoir computers. Each reservoir is then used to make a prediction and a VPT is calculated. This process is repeated for both local and global predictions on each attractor.

The results of these experiments are shown in Figure 11 for each attractor (right to left) and for global and local predictions (top to bottom). For each system, training method, and prediction type, the mean VPT is plotted (solid line) and $\pm 1$ standard deviation (shaded region) across the training lengths we consider.

For the global prediction task, the windowed method outperforms both the standard and partial method with increasing accuracy as the training time is increased for the Lorenz and Rossler attractor over the range we consider. For each system, the partial method is only marginally better than the standard method for most training times with the exception of the Thomas attractor, where the partial method lies directly between the other two methods in terms of accuracy. For the Lorenz attractor, the average VPTs seem to flatten out as training times increase suggesting an asymptotic maximum accuracy for this system, where further training does not significantly increase precision. This trend may also hold for the Thomas attractor. Still, both may have an optimal training time like the Rossler attractor which maxes out near $T = 500$, but much larger simulations may be needed to determine if this is the case.

For the local prediction task there is less agreement. For the Lorenz attractor, the windowed method outperforms the other

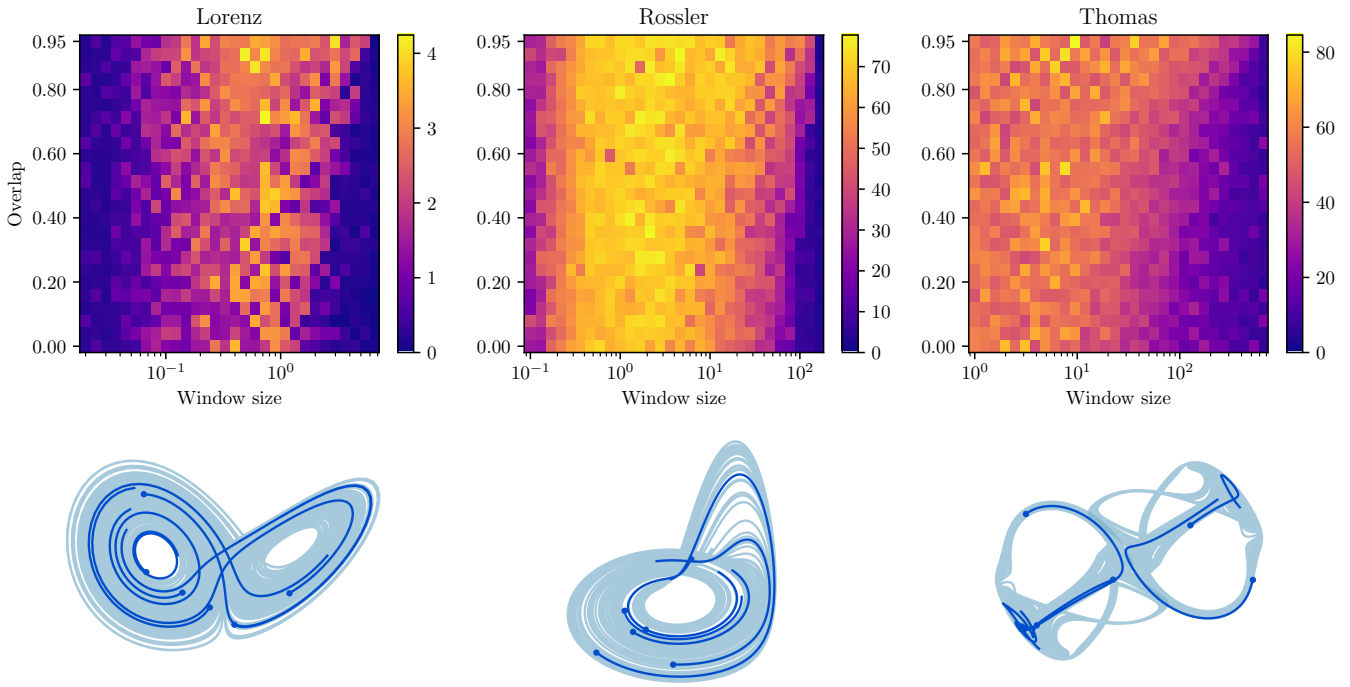


FIG. 12. (Top) Comparison of the VPT of the windowed method on the global prediction task as the window size $L$ and overlap $v$ are varied. The distribution of VPTs is roughly unimodal with respect to $L$, with peaks at approximately $L = 1$ for the Lorenz system, $L = 3$ for the Rossler, and $L = 10$ for the Thomas. The choice of overlap $v$ has less influence on the VPT for smaller window sizes, but for larger window sizes a higher value of $v$ results in higher VPTs. (Bottom) For each system we show eight signals of length $L = 1$ on the Lorenz system, $L = 3$ for the Rossler, and $L = 10$ for the Thomas to illustrate the optimal window sizes on the respective attractors.

methods for the large majority of the training times we consider having a near unimodal shape with a maximum around $T = 10$. For the Rossler attractor, a similar pattern occurs, with the maximum around $T = 500$. The same pattern may occur for the Thomas attractor if a larger range of training times are considered. On the Rossler system the three methods perform similarly for short training times, but the windowed method outperforms the other two for longer signal lengths. On the Thomas system, all three methods perform similarly, with the partial method slightly outperforming the others.

### E. Dependence on Window Size and Overlap

Finally, we seek to investigate how the choice of window length $L > 0$ and overlap $v \geq 0$ affect the accuracy of reservoir computers trained with the windowed method on the global prediction task. For each attractor, we select a grid of 30 $L$-values for window size and 25 $v$-values for overlap, with overlap spaced linearly and window size spaced logarithmically. These experiments use the following parameters, with a different set of window sizes for each attractor:

$$\begin{aligned} L_{Lorenz} &\in [0.02, 6.6], \\ L_{Rossler} &\in [0.1, 165], \\ L_{Thomas} &\in [1, 660], \\ v &\in [0, 0.95], \\ n &= 1000, \\ c &= 1.0. \end{aligned}$$

The range of window lengths is selected so that each window is at least twice the time step size used, and at most the training signal length for each system. BHO is used to select the other hyperparameters, with 50 hyperparameter choices tested with $2^8$ trials each. The displayed VPTs for each experiment are the average of $2^9$ trials with the best found hyperparameters.

In Figure 12, we summarize the results of these experiments. In each pixel, the average VPT of the experiment is indicated by its color. Yellow indicates higher VPTs, while purple indicates lower values.

Several trends are visible in the data, which are very similar between the three attractors. First, the VPT is much more variable with respect to the window size $L$; for most $L$-values. Changing the overlap $v$ has very little influence on the VPT. The exception is that for larger window sizes, higher overlap values result in higher VPTs. On each of the attractors, the VPT is approximately unimodal with respect to the window size. As $L \to 0$ and $L \to T$, the valid prediction time decreases on the Lorenz and Rossler systems. In both of these systems,

the VPT typically decreases to zero. On the Thomas system, the VPT decreases as $L$ increases, although less dramatically compared to the other systems, and reducing $L$ does not decrease the VPT by much for the parameter ranges we consider. The peak VPT occurs for window size approximately $L_{Lorenz} = 1$ on the Lorenz attractor, around $L_{Rossler} = 3$ on the Rossler attractor, and around $L_{Thomas} = 10$ on the Thomas attractor. These are potentially related to properties of these attractors, such as Lyapunov exponents and time-around-the-attractor, but the precise correspondence is unclear.

To visualize the latter, in Figure 12(bottom) we show the distance across the attractor for these approximately optimal window sizes. For the Lorenz attractor, trajectories of length $L = 1$ can move across roughly two-thirds of the attactor. For the Rossler attractor, trajectories of length $L = 3$ move at most through one-half of the attractor. On the Thomas attractor trajectories move through at most half of the right or left side of the attractor depending on which side they start on although some trajectories are much shorter. This gives us at least a sense for the scale at which an attractor is best learned by a reservoir using the windowed method.

## VII. PREDICTION ERROR BOUNDS

In this section, we prove several error bounds in terms of (9) and (10), discussed in Section II. These bounds demonstrate that these two conditions are sufficient for a reservoir computer to create accurate predictions. This in particular helps us to understand why the windowed method can effectively create global predictions, while the standard method cannot.

We begin by formulating these conditions more precisely. We will use our notation for the global forecasting task; however, everything in this section is also applicable to local predictions. Suppose we are trying to predict the signal $\mathbf{v}(t)$ for $t \geq T$ with $\mathbf{v}(T) = \mathbf{v}$. Our reservoir computer is trained on $\mathbf{u}(t)$ for $t \in [0,T]$ and creates the prediction $\hat{\mathbf{v}}(t) = \mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t)$ of $\mathbf{v}(t)$, where $\hat{\mathbf{r}}(T) = \hat{\mathbf{r}}_T$. Define the constants $E_I$ and $E_D$ as follows:

$$E_I = \|\mathbf{W}_{\text{out}}\hat{\mathbf{r}}_T - \mathbf{v}\| \tag{15}$$

$$E_D = \sup_{t \geq T} \|\mathbf{W}_{\text{out}} h(\hat{\mathbf{r}}(t)) - g(\mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t))\| \tag{16}$$

The constant $E_I$ represents the error associated with condition (9) regarding the initial condition of the reservoir computer. The constant $E_D$ represents a bound on the error relative to condition (10) for the reservoir's approximation of the dynamics of the dynamical system. The bounds we discuss can be made more precise by replacing $E_D$ with the exact value of $\|\mathbf{W}_{\text{out}} h(\hat{\mathbf{r}}(t)) - g(\mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t))\|$ as appropriate, but for simplicity the analysis here will not do so.

In terms of these constants, we can give several bounds on the growth of the *prediction error* given by

$$||\hat{\mathbf{v}}(t) - \mathbf{v}(t)|| \text{ for } t \geq T.$$

These can in turn be used to give lower bounds on the valid prediction time.

First, supposing only that $g$ is Lipschitz continuous, we have the following simple bound.

**Theorem 1.** ***(Lipschitz Error-Bound)*** *Let $g$ and $h$ be defined as in Equations* (7) *and* (8)*, respectively. Also, let $E_I$ and $E_D$ be defined as in Equations* (15) *and* (16)*, respectively. Suppose that $g$ is Lipschitz continuous on $A$ with Lipschitz constant $L$. Then, the prediction error satisfies*

$$\|\hat{\mathbf{v}}(t) - \mathbf{v}(t)\| \leq (E_D t + E_I)e^{Lt}.$$

If $E_I$ and $E_D$ are small then we have a relatively good bound on our approximation error, although it grows exponentially in time relative to the Lipschitz constant $L$. With the bound of this theorem, it is clear that as both $E_I \to 0$ and $E_D \to 0$, the error $\|\mathbf{v}(t) - \hat{\mathbf{v}}(t)\|_2 \to 0$ pointwise. This demonstrates that in order to reduce the prediction error, it suffices to reduce the initial error $E_I$ and dynamics error $E_D$ of the reservoir computer. However, this bound is very pessimistic and does not give us a good idea of how the prediction error typically grows (cf Figure 13).

For a sharper bound we can use a function related to the dynamical system's Lyapunov exponent on its attractor to achieve the following. Define the function $\Lambda : A \times A \to \mathbb{R}$ by

$$\Lambda(\mathbf{v}_1, \mathbf{v}_2) = \begin{cases} \frac{\langle \mathbf{v}_1 - \mathbf{v}_2,\, g(\mathbf{v}_1) - g(\mathbf{v}_2)\rangle}{\|\mathbf{v}_1 - \mathbf{v}_2\|_2^2} & \mathbf{v}_1 \neq \mathbf{v}_2 \\ 0 & \mathbf{v}_1 = \mathbf{v}_2 \end{cases} \tag{17}$$

where $\langle \cdot, \cdot \rangle$ is the standard inner product on $\mathbb{R}^m$ and $\mathbf{v}_1$ and $\mathbf{v}_2$ are arbitrary points in $A$. The function $\Lambda$ measures how quickly trajectories beginning at two points on $A$ diverge from each other. Specifically, if $\mathbf{v}_1(t), \mathbf{v}_2(t)$ are trajectories on $A$ then $\Lambda$ satisfies

$$\frac{d}{dt}\|\mathbf{v}_1(t) - \mathbf{v}_2(t)\|_2 = \Lambda(\mathbf{v}_1(t), \mathbf{v}_2(t))\|\mathbf{v}_1(t) - \mathbf{v}_2(t)\|_2 \tag{18}$$

and

$$\Lambda(\mathbf{v}_1(t), \mathbf{v}_2(t)) = \frac{d}{dt}\ln\|\mathbf{v}_1(t) - \mathbf{v}_2(t)\|_2. \tag{19}$$

This is proven in Proposition X1 in the Appendix. In terms of the function $\Lambda$ we have the following tighter bound:

**Theorem 2.** ***(Lyapunov Error-Bound)*** *For the input-signal $\mathbf{v}(t)$ where $t \geq T$ and prediction $\hat{\mathbf{v}}(t)$ let*

$$V(t) = \int_T^t \Lambda(\mathbf{v}(s), \hat{\mathbf{v}}(s))\, ds.$$

*Suppose $g$ and $h$ are Lipschitz continuous on $A$ and $R$, respectively. With $E_I$ and $E_D$ given by Equations* (15) *and* (16)*, respectively we have the error bound*

$$\|\hat{\mathbf{v}}(t) - \mathbf{v}(t)\|_2 \leq \exp\big(V(t)\big)\left(E_I + E_D\int_T^t \exp\big(-V(s)\big)\, ds\right)$$

*for all $t \geq T$.*

The proofs of Theorems 1 and 2 are given in the Appendix. To make the bound of Theorem 2 more concrete, we give an approximate form using the principle (largest) Lyapunov exponent $\lambda$ of the dynamical system. Let $\mathbf{v}_1(t), \mathbf{v}_2(t)$ be trajectories of the dynamical system with $\mathbf{v}_2(0) = \mathbf{v}_1(0) + \delta\mathbf{v}$ where $\delta > 0$. Using equation (19), we can then write the formula for a Lyapunov exponent[35] as

$$\begin{aligned}\lambda &= \lim_{t\to\infty} \lim_{\|\delta\mathbf{v}\|\to 0} \frac{1}{t} \ln \frac{\|\mathbf{v}_1(t) - \mathbf{v}_2(t)\|}{\|\delta\mathbf{v}\|} \\ &= \lim_{t\to\infty} \lim_{\|\delta\mathbf{v}\|\to 0} \frac{1}{t} \int_0^t \Lambda(\mathbf{v}_1(s), \mathbf{v}_2(s))\, ds.\end{aligned}$$

Generally, this will converge to the largest Lyapunov exponent. For sufficiently large $t$ and sufficiently small $\delta\mathbf{v}$, we can approximate the limit by truncating it, giving the approximation

$$\int_0^t \Lambda(\mathbf{v}_1(s), \mathbf{v}_2(s))\, ds \approx \lambda t. \tag{20}$$

Applying this to the bound from Theorem 2 we have, for sufficiently large $t$ and sufficiently small $E_I$, the bound

$$\|\hat{\mathbf{v}}(t) - \mathbf{v}(t)\|_2 \lessapprox e^{\lambda t}\left(E_I + \frac{E_D}{\lambda}\right) - \frac{E_D}{\lambda}. \tag{21}$$

If $\lambda > 0$ this bound has the same order as measuring the distance between different trajectories of the underlying dynamical system as they diverge from each other. Since some amount of error is inevitable this is thus the best order of error we can expect. So, by designing the reservoir computer to satisfy conditions (9) and (10) as precisely as possible, we not only will reduce the prediction error, but can expect the prediction error to grow with $t$ as slowly as can be reasonably expected.

In Figure 13, we give an example of these error bounds for the reservoir computer from Example 1.

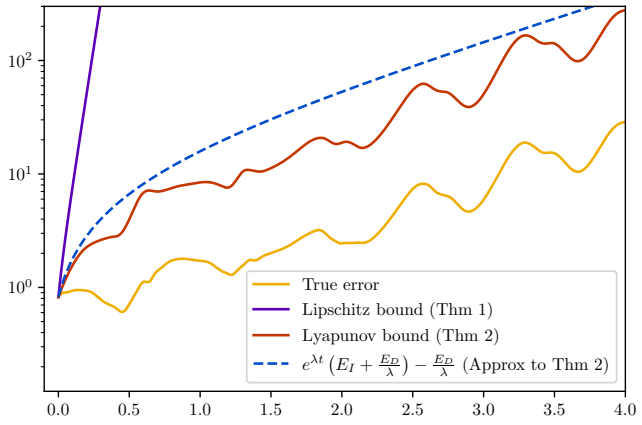


FIG. 13. The true error (in the 2-norm) of the prediction of the reservoir computer in Example 1 (yellow), alongside the bounds from Theorems 1 (purple) and 2 (red), as well as the approximate form to Theorem 2 (dashed black). For this example, we have $E_I = 0.82256$ and $E_D = 8.48682$.

## VIII. CONCLUSION

The goal of this paper is to understand to what extent it is possible to create a global forecast of a dynamical system based on a single-input signal. While we show this to be possible for a number of chaotic dynamical systems, this goes beyond the standard use of training computer reservoirs to create a local forecast of an input-signal. As local forecasts can be used to reproduce features of a chaotic system such as first-return maps, Lyapunov exponents, predict bifurcations, phase, and so forth, it is an open question as to whether using the windowed method can more reliably recreate these quantities by sampling over the system's full attractor. Currently, it is unknown to what extent the use of windows may be useful in this endeavor. Moreover, it is not clear at this time how other methods compare in accuracy to the windowed method including the method proposed in (Griffith et al., 2019)[24] with regard to these tasks.

The windowed method can also be used to make predictions on systems that have no known set of underlying equations. Although our focus in this paper is on chaotic systems given by specific sets of equations, the windowed method can be equally well used on input-signals derived from experimental data, etc. As an example, in a following paper we plan to explore how controlled systems can be learned at the global level using the windowed method directly on time-series data. At the present it is not well-understood how accurate a global forecast of such systems can be, but our method can forecast from many different initial conditions potentially giving us insight into understanding such systems.

In terms of the specific results of this paper, one of the less intuitive findings is that the accuracy of the windowed method is system dependent. In our experiments, the windowed method gives improved accuracy for local predictions on the Lorenz attractor but not on the Rossler or Thomas attractors. The optimal training times for global to local predictions differ for the Lorenz attractor but are nearly the same for the Rossler attractor. Also, optimal window sizes differ from attractor to attractor and it is unknown to what extent these window sizes, training times, and differences in local/global accuracy is related to specific features of these attractors.

There are also unknowns about other reservoir features such as consistency[36] and robustness to noise. Specifically, it is unknown whether the windowed method interacts favorably with these features creating a reservoir that is *globally* consistent and/or robust to noise over the system's attractor or whether these properties vary wildly from initial condition to initial condition.

## IX. APPENDIX

Here we prove the error bounds found in Section VII, as well as several properties of the function $\Lambda$.

**Proof of Theorem 1.** We can write the prediction error as

$$\begin{aligned}
\|\hat{\mathbf{v}}(t)-\mathbf{v}(t)\| &= \left\| \int_0^t (\hat{\mathbf{v}}'(s)-\mathbf{v}'(s))\,ds + \hat{\mathbf{v}}(0)-\mathbf{v}(0) \right\| \\
&\le \int_0^t \left\|\hat{\mathbf{v}}'(s)-\mathbf{v}'(s)\right\| ds + E_I \\
&= \int_0^t \|\mathbf{W}_{\text{out}} h(\hat{\mathbf{r}}(s)) - g(\mathbf{v}(s))\|\,ds + E_I \\
&\le \int_0^t \|\mathbf{W}_{\text{out}} h(\hat{\mathbf{r}}(s)) - g(\mathbf{W}_{\text{out}}\hat{\mathbf{r}}(s))\|\,ds \\
&\quad + \int_0^t \|g(\mathbf{W}_{\text{out}}\hat{\mathbf{r}}(s)) - g(\mathbf{v}(s))\|\,ds + E_I \\
&\le \int_0^t \|g(\mathbf{W}_{\text{out}}\hat{\mathbf{r}}(s)) - g(\mathbf{v}(s))\|\,ds + tE_d + E_I \\
&\le L\int_0^t \|\hat{\mathbf{v}}(s)-\mathbf{v}(s)\|\,ds + tE_D + E_I.
\end{aligned}$$

Applying Gronwall's inequality[37] (Theorem 1.3.1) gives the desired bound. □

**Proposition X1.** *Suppose that g is Lipschitz continuous with Lipschitz constant L and $\mathbf{u}_1(t), \mathbf{u}_2(t)$ are trajectories of a dynamical system. Then, the function $\Lambda$ (see Equation (17)) has the following properties:*

1. $|\Lambda(\mathbf{u},\mathbf{v})| \le L$ *for all* $\mathbf{u},\mathbf{v} \in A$;
2. $\frac{d}{dt}\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\|_2 = \Lambda(\mathbf{u}_1(t),\mathbf{u}_2(t))\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\|_2$;
3. $\Lambda(\mathbf{u}_1(t),\mathbf{u}_2(t)) = \frac{d}{dt}\ln\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\|_2$.

*Proof.* Property (1) is clear if $\mathbf{u}=\mathbf{v}$. For $\mathbf{u}\neq\mathbf{v}$, it follows immediately from the Cauchy-Schwartz inequality:

$$\begin{aligned}
|\Lambda(\mathbf{u},\mathbf{v})| &= \left| \frac{\langle \mathbf{u}-\mathbf{v}, g(\mathbf{u})-g(\mathbf{v})\rangle}{\|\mathbf{u}-\mathbf{v}\|_2^2} \right| \\
&\le \frac{\|\mathbf{u}-\mathbf{v}\|_2 \|g(\mathbf{u})-g(\mathbf{v})\|_2}{\|\mathbf{u}-\mathbf{v}\|_2^2} \\
&\le L.
\end{aligned}$$

To prove Property (3), we compute that

$$\begin{aligned}
&\frac{d}{dt}\ln\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\| \\
&= \frac{1}{2}\frac{d}{dt}\ln\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\|^2 \\
&= \frac{\frac{d}{dt}\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\|^2}{2\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\|^2} \\
&= \frac{2\left\langle \mathbf{u}_1(t)-\mathbf{u}_2(t), \frac{d}{dt}(\mathbf{u}_1(t)-\mathbf{u}_2(t))\right\rangle}{2\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\|^2} \\
&= \frac{\langle \mathbf{u}_1(t)-\mathbf{u}_2(t), g(\mathbf{u}_1(t))-g(\mathbf{u}_2(t))\rangle}{\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\|^2}.
\end{aligned}$$

Finally, property (2) follows immediately from property (3) with the observation that

$$\frac{d}{dt}\ln\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\| = \frac{\frac{d}{dt}\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\|}{\|\mathbf{u}_1(t)-\mathbf{u}_2(t)\|}.$$

□

To prove Theorem 2, we will use the following lemma.

**Lemma X2.** *Let $f(t,x) : [0,\infty)\times\mathbb{R}\to\mathbb{R}$ be a continuous function that is Lipschitz continuous with respect to x uniformly in t. If $u,v \in C^1([0,\infty),\mathbb{R})$ satisfy*

$$u'(t) \le f(t,u(t)), \quad v'(t) = f(t,v(t)), \quad u(0)\le v(0)$$

*for all $t\ge 0$, then $u(t)\le v(t)$ for all $t\ge 0$.*

**Proof.** First suppose that $u'(t) < f(t,u(t))$ for all $t$. Let $w(t) = v(t)-u(t)$. Our claim is equivalent to showing that $w(t)\ge 0$ for all $t\ge 0$. Suppose to the contrary that there exists $t_0$ such that $w(t_0)<0$. Let

$$t^* = \inf\{t\ge 0 : w(t) < 0\},$$

which exists because the set is nonempty and bounded below. Since $w(0)\ge 0$, we have by continuity that $w(t^*)=0$ as well, or $u(t^*)=v(t^*)$. Since $w(t)<0$ for $t\in(t^*,t^*+\delta)$ for some $\delta>0$, it must be that $w'(t^*)\le 0$. However, by assumption,

$$w'(t^*) = v'(t^*)-u'(t^*) > f(t^*,v(t^*)) - f(t^*,u(t^*)) = 0,$$

a contradiction. This shows the lemma in the case that $u'(t) < f(t,u(t))$.

We now proceed to the general case where we assume $u'(t)\le f(t,u(t))$. For any $\varepsilon>0$ we have

$$u'(t) \le f(t,u(t)) < f(t,u(t)) + \varepsilon = g_\varepsilon(t,u(t)).$$

Define $v_\varepsilon(t)$ by

$$v_\varepsilon(0) = v(0), \quad v_\varepsilon'(t) = g_\varepsilon(t,v_\varepsilon(t)).$$

Since $f$ is Lipschitz, so is $g_\varepsilon$. Thus each $v_\varepsilon$ is well-defined, and we can apply the special case proved above to obtain $u(t)\le v_\varepsilon(t)$ for all $t\ge 0$, $\varepsilon>0$.

We finish by showing that $v_\varepsilon \to v$ pointwise as $\varepsilon\to 0$, which shows the desired bound. Suppose that $f$ has Lipschitz constant $L$ with respect to its second argument; that is,

$$|f(t,x)-f(t,y)| \le L|x-y|$$

for all $t\in[0,\infty)$ and $x,y\in\mathbb{R}$. Then,

$$\begin{aligned}
|v_\varepsilon(t)-v(t)| &= \left| \int_0^t v_\varepsilon'(s)-v'(s)\,ds \right| \\
&\le \int_0^t \left| v_\varepsilon'(s)-v'(s)\,ds \right| \\
&= \int_0^t |g_\varepsilon(t,v_\varepsilon(s)) - f(t,v(s))\,ds| \\
&\le \int_0^t |f(t,v_\varepsilon(s)) - f(t,v(s))|\,ds + t\varepsilon \\
&\le L\int_0^t |v_\varepsilon(s)-v(s)|\,ds + t\varepsilon.
\end{aligned}$$

Applying Gronwall's inequality[37] (Theorem 1.3.1) implies that

$$|v_\varepsilon(t)-v(t)| \le t\varepsilon e^{Lt} \to 0$$

pointwise as $\varepsilon \to 0$. Thus

$$u(t) \le \lim_{\varepsilon\to 0} v_\varepsilon(t) = v(t)$$

completing the proof. □

It should be noted that this lemma also holds if $u'$ is taken to denote the right-hand-side derivative of $u$.

**Proof of Theorem 2.** Let $\|\cdot\|$ denote the 2-norm. We proceed by showing that

$$\frac{d}{dt}\|\hat{\mathbf{v}}(t)-\mathbf{v}(t)\| \le \Lambda(\mathbf{v}(t),\hat{\mathbf{v}}(t))\|\hat{\mathbf{v}}(t)-\mathbf{v}(t)\| + E_D$$

and then applying the lemma. Although $\|\hat{\mathbf{v}}(t)-\mathbf{v}(t)\|$ may not be differentiable when it is zero, the right-hand-side derivative will always exist, which is all we need for the lemma to apply.

Let the flow function of the dynamical system be denoted $U(t,\mathbf{v})$.

We have

$$\begin{aligned}
&\frac{d}{dt}\|\hat{\mathbf{v}}(t)-\mathbf{v}(t)\| \\
&\quad= \lim_{\Delta t\to 0^+}\frac{1}{\Delta t}\Big(\|\hat{\mathbf{v}}(t+\Delta t)-\mathbf{v}(t+\Delta t)\| - \|\hat{\mathbf{v}}(t)-\mathbf{v}(t)\|\Big) \\
&\quad\le \lim_{\Delta t\to 0^+}\frac{1}{\Delta t}\Big(\|U(\Delta t,\mathbf{v}(t))-U(\Delta t,\hat{\mathbf{v}}(t))\| \\
&\qquad + \|\hat{\mathbf{v}}(t+\Delta t)-U(\Delta t,\mathbf{v}(t))\| - \|\hat{\mathbf{v}}(t)-\mathbf{v}(t)\|\Big). \quad \text{(A1)}
\end{aligned}$$

First, observe that since

$$\begin{aligned}
&\frac{d}{ds}\|U(s,\mathbf{v}(t)),U(s,\hat{\mathbf{v}}(t))\| \\
&\quad= \Lambda(U(s,\mathbf{v}(t)),U(s,\hat{\mathbf{v}}(t)))\|U(s,\mathbf{v}(t)),U(s,\hat{\mathbf{v}}(t))\|,
\end{aligned}$$

we have

$$\begin{aligned}
&\|U(\Delta t,\mathbf{v}(t))-U(\Delta t,\hat{\mathbf{v}}(t))\| \\
&\quad= \Big(1+\Lambda\big(U(0,\mathbf{v}(t)),U(0,\hat{\mathbf{v}}(t))\big)\Delta t + O(\Delta t^2)\Big) \\
&\qquad\cdot \|U(0,\mathbf{v}(t))-U(0,\hat{\mathbf{v}}(t))\| \\
&\quad= \big(1+\Delta t\Lambda(\mathbf{v}(t),\hat{\mathbf{v}}(t))+O(\Delta t^2)\big)\|\mathbf{v}(t)-\hat{\mathbf{v}}(t)\|. \quad \text{(A2)}
\end{aligned}$$

Denote the Lipschitz constants of $g$ and $h$ as $L_1$ and $L_2$, respectively. Then

$$\begin{aligned}
&\|\hat{\mathbf{v}}(t+\Delta t)-U(\Delta t,\hat{\mathbf{v}}(t))\| \\
&\quad\le \int_t^{t+\Delta t}\big\|\hat{\mathbf{v}}'(s)-U_t(s,\hat{\mathbf{v}}(t))\big\|\,ds \\
&\quad= \int_t^{t+\Delta t}\|\mathbf{W}_{\text{out}}h(\hat{\mathbf{r}}(s))-g(U(s,\hat{\mathbf{v}}(t)))\|\,ds \\
&\quad= \int_t^{t+\Delta t}\|\mathbf{W}_{\text{out}}h(\hat{\mathbf{r}}(t)+O(\Delta t))-g(\hat{\mathbf{v}}(t)+O(\Delta t))\|\,ds \\
&\quad\le \int_t^{t+\Delta t}\big(\|\mathbf{W}_{\text{out}}h(\hat{\mathbf{r}}(t)))-g(\hat{\mathbf{v}}(t))\| \\
&\qquad + \|\mathbf{W}_{\text{out}}\|L_1O(\Delta t)+L_2O(\Delta t)\big)\,ds \\
&\quad= \int_t^{t+\Delta t}\|\mathbf{W}_{\text{out}}h(\hat{\mathbf{r}}(t)))-g(\mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t))\|\,ds + O(\Delta t^2) \\
&\quad\le E_D\Delta t + O(\Delta t^2). \quad \text{(A3)}
\end{aligned}$$

Combining (A2) and (A3) with (A1) gives that

$$\begin{aligned}
&\frac{d}{dt}\|\hat{\mathbf{v}}(t)-\mathbf{v}(t)\| \\
&\quad\le \lim_{\Delta t\to 0^+}\frac{1}{\Delta t}\big(\|U(\Delta t,\mathbf{v}(t))-U(\Delta t,\hat{\mathbf{v}}(t))\| \\
&\qquad + \|\hat{\mathbf{v}}(t+\Delta t)-U(\Delta t,\mathbf{v}(t))\| - \|\hat{\mathbf{v}}(t)-\mathbf{v}(t)\|\big) \\
&\quad= \lim_{\Delta t\to 0^+}\frac{1}{\Delta t}\Big(\big(\big(1+\Delta t\Lambda(\mathbf{v}(t),\hat{\mathbf{v}}(t))+O(\Delta t^2)\big)\|\mathbf{v}(t)-\hat{\mathbf{v}}(t)\|\big) \\
&\qquad + \big(E_D\Delta t+O(\Delta t^2)\big) - \|\hat{\mathbf{v}}(t)-\mathbf{v}(t)\|\Big) \\
&\quad= \Lambda(\mathbf{v}(t),\hat{\mathbf{v}}(t))\|\mathbf{v}(t)-\hat{\mathbf{v}}(t)\| + E_D.
\end{aligned}$$

Finally, applying the lemma to $\|\hat{\mathbf{v}}(t)-\mathbf{v}(t)\|$ with $f$ given by

$$f(t,y) = \Lambda(\mathbf{v}(t),\hat{\mathbf{v}}(t))y + E_D$$

gives the result. □

## X. AUTHOR DECLARATIONS

Conflict of Interest: The authors have no conflicts to disclose.

## XI. DATA AVAILABILITY

The data that support the findings of this study and the source code to reproduce the presented results are available from the corresponding author upon reasonable request.

## FUNDING

B.Z.W. was partially supported by the NSF grant #2205837 in Applied Mathematics.

## REFERENCES

[1] Y. Chen, Y. Qian, and X. Cui, "Time series reconstructing using calibrated reservoir computing," Scientific Reports **12** (2022).

[2] L.-W. Kong, H.-W. Fan, C. Grebogi, and Y.-C. Lai, "Machine learning prediction of critical transition and system collapse," Phys. Rev. Res. **3**, 013090 (2021).

[3] C. Zhang, J. Jiang, S.-X. Qu, and Y.-C. Lai, "Predicting phase and sensing phase coherence in chaotic systems with machine learning." Chaos **30 8**, 083114 (2020).

[4] S. Krishnagopal, M. Girvan, E. Ott, and B. R. Hunt, "Separation of chaotic signals by reservoir computing," Chaos **30 2**, 023123 (2019).

[5] T. L. Carroll, "Using reservoir computers to distinguish chaotic signals," Physical Review E (2018).

[6] Z. Lu, J. Pathak, B. R. Hunt, M. Girvan, R. W. Brockett, and E. Ott, "Reservoir observers: Model-free inference of unmeasured variables in chaotic systems." Chaos **27 4**, 041102 (2017).

[7] K. Nakai and Y. Saiki, "Machine-learning inference of fluid variables from data using reservoir computing." Physical review. E **98 2-1**, 023111 (2018).

[8] J. Pathak, B. R. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach." Physical review letters **120 2**, 024102 (2018).

[9]P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," Neural networks : the official journal of the International Neural Network Society **126**, 191–217 (2019).

[10]H. 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** (2001).

[11]M. H. Maass W, Natschläger T, "Real-time computing without stable states: a new framework for neural computation based on perturbations." Neural Computation **14**, 2531–2560 (2002).

[12]Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," Chaos: An Interdisciplinary Journal of Nonlinear Science **28**, 061104 (2018), https://doi.org/10.1063/1.5039508.

[13]N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a time series," Phys. Rev. Lett. **45**, 712–716 (1980).

[14]F. Takens, "Detecting strange attractors in turbulence. dynamical systems and turbulence," in *Dynamical Systems and Turbulence, Warwick 1980* (Springer, Berlin, Heidelberg, 1981) pp. 366–381.

[15]T. Sauer, J. A. Yorke, and M. Casdagli, "Embedology," Journal of Statistical Physics **65**, 579–616 (1991).

[16]R. Yu, S. Zheng, and Y. Liu, "Learning chaotic dynamics using tensor recurrent neural networks," (2017).

[17]M. Goldmann, I. Fischer, C. R. Mirasso, and M. C. Soriano, "Exploiting oscillatory dynamics of delay systems for reservoir computing," Chaos: An Interdisciplinary Journal of Nonlinear Science **33**, 093139 (2023), https://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0156494/18139000/093139_1_5.0156494.pdf.

[18]K. Aihara, T. Takabe, and M. Toyoda, "Chaotic neural networks," Physics Letters A **144**, 333 – 340 (1990).

[19]M. Sangiorgio and F. Dercole, "Robustness of lstm neural networks for multi-step forecasting of chaotic time series," Chaos, Solitons & Fractals **139**, 110045 (2020).

[20]P. Vlachas, J. Pathak, B. Hunt, T. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," Neural Networks **126**, 191 – 217 (2020).

[21]J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data." Chaos **27 12**, 121102 (2017).

[22]J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," Phys. Rev. Lett. **120**, 024102 (2018).

[23]J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model," Chaos: An Interdisciplinary Journal of Nonlinear Science **28**, 041101 (2018), https://doi.org/10.1063/1.5028373.

[24]A. Griffith, A. Pomerance, and D. J. Gauthier, "Forecasting chaotic systems with very low connectivity reservoir computers," Chaos: An Interdisciplinary Journal of Nonlinear Science **29**, 123108 (2019), https://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/1.5120710/14623672/123108_1_online.pdf.

[25]J. A. Platt, S. G. Penny, T. A. Smith, T.-C. Chen, and H. D. Abarbanel, "A systematic exploration of reservoir computing for forecasting complex spatiotemporal dynamics," Neural Networks **153**, 530–552 (2022).

[26]S. G. Penny, T. A. Smith, T.-C. Chen, J. A. Platt, H.-Y. Lin, M. R. Goodliff, and H. D. I. Abarbanel, "Integrating recurrent neural networks with data assimilation for scalable data-driven state estimation," Journal of Advances in Modeling Earth Systems **14** (2021).

[27]G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," Neural Networks **115**, 100 – 123 (2019).

[28]N. Lorenz, "Deterministic nonperiodic flow," J. Atmospheric Science **20**, 130–141 (1962).

[29]O. Rössler, "An equation for continuous chaos," Physics Letters A **57**, 397–398 (1976).

[30]R. THOMAS, "Deterministic chaos seen in terms of feedback circuits: Analysis, synthesis, "labyrinth chaos"," International Journal of Bifurcation and Chaos **09**, 1889–1905 (1999), https://doi.org/10.1142/S0218127499001383.

[31]J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," Journal of Electronic Science and Technology **17**, 26 (2019).

[32]D. Passey, Q. Leishman, T. Pool, S. Harding, and W. Lunceford, "RCInitialCond," `https://github.com/djpasseyjr/RCInitialCond` (2023).

[33]D. Passey and Q. Leishman, "Rescomp," `https://github.com/djpasseyjr/Rescomp` (2020).

[34]L. Hertel, J. Collado, P. Sadowski, J. Ott, and P. Baldi, "Sherpa: Robust hyperparameter optimization for machine learning," SoftwareX (2020), in press.

[35]M. Cencini, F. Cecconi, and A. Vulpiani, *Chaos: From Simple Models to Complex Systems*, Series on advances in statistical mechanics (World Scientific, 2010).

[36]T. Lymburn, A. Khor, T. Stemler, D. Correa, M. Small, and T. Jungling, English"Consistency in echo-state networks," Chaos: an interdisciplinary journal of nonlinear science **29**, 1–9 (2019).

[37]B. G. Pachpatte, *Inequalities for differential and integral equations* (Academic Press, San Diego, California, 1998).