Reservoir observers: Model-free inference of unmeasured variables in chaotic systems

Zhixin Lu, Jaideep Pathak, Brian Hunt, Michelle Girvan, Roger Brockett, and Edward Ott

Citation: Chaos **27**, 041102 (2017); doi: 10.1063/1.4979665
View online: http://dx.doi.org/10.1063/1.4979665
View Table of Contents: http://aip.scitation.org/toc/cha/27/4
Published by the American Institute of Physics

Reservoir observers: Model-free inference of unmeasured variables in chaotic systems

Zhixin Lu,[1,2] Jaideep Pathak,[1] Brian Hunt,[2] Michelle Girvan,[1,2] Roger Brockett,[3] and Edward Ott[1]
[1]*Institute for Research in Electronics and Applied Physics, University of Maryland, College Park, Maryland 20742, USA*
[2]*Institute for Physical Science and Technology, University of Maryland, College Park, Maryland 20742, USA*
[3]*John A. Paulson School of Engineering and Applied Science, Harvard University, Cambridge, Massachusetts 02138, USA*

(Received 22 February 2017; accepted 22 March 2017; published online 5 April 2017)

Deducing the state of a dynamical system as a function of time from a limited number of concurrent system state measurements is an important problem of great practical utility. A scheme that accomplishes this is called an "observer." We consider the case in which a model of the system is unavailable or insufficiently accurate, but "training" time series data of the desired state variables are available for a short period of time, and a limited number of other system variables are continually measured. We propose a solution to this problem using networks of neuron-like units known as "reservoir computers." The measurements that are continually available are input to the network, which is trained with the limited-time data to output estimates of the desired state variables. We demonstrate our method, which we call a "reservoir observer," using the Rössler system, the Lorenz system, and the spatiotemporally chaotic Kuramoto–Sivashinsky equation. Subject to the condition of observability (i.e., whether it is in principle possible, by any means, to infer the desired unmeasured variables from the measured variables), we show that the reservoir observer can be a very effective and versatile tool for robustly reconstructing unmeasured dynamical system variables. *Published by AIP Publishing.* [http://dx.doi.org/10.1063/1.4979665]

Knowing the state of a dynamical system as it evolves in time is important for a variety of applications. This paper proposes a general-purpose method for inferring unmeasured state variables from a limited set of ongoing measurements. Our method is intended for situations in which mathematical models of system dynamics are unavailable or are insufficiently accurate to perform the desired inference. We use the machine-learning technique called "reservoir computing," with which we construct a system-independent means of processing the measurements. A key point is the extent to which this approach is "universal." That is, our examples show that the same reservoir can be trained to infer the state of different systems. It is the training that relates to a specific system, not the "hardware." The reservoir hardware plays a similar role to an animal's brain, which retrains itself as the system represented by its body and environment changes.

I. INTRODUCTION

Frequently, when studying the dynamics of a physical system, one only has access to a limited set of measurements of the state variables and desires to deduce concurrent values of unmeasured state variables. In principle, it might be possible to accomplish this goal if, in addition to the measurements, one also has knowledge of the system dynamics. In control theory, a successful deduction method of this type is called an *observer*. Observers are of great utility for control and prediction of dynamics. The observer problem for the case in which the dynamical system is linear was fully solved in the classic work of Kalman, who also formulated conditions for "observability" under which it is possible to achieve the goal of deducing the full state of a linear system from a given partial set of state measurements (see textbooks on control theory, e.g., Ref. 1). Observers and observability have also been extensively investigated for nonlinear dynamical systems (e.g., Ref. 2). For example, in situations of chaotic dynamics, an approach using synchronization of chaos has been exploited.[3,4] A concept related to observability, called "estimability" has been discussed in the context of delay coordinate embedding.[5]

In this paper we consider the observer problem for situations in which one does not have a sufficiently accurate mathematical model of the nonlinear system of interest. In place of such a model, we assume that there exists an initial period of time for which measurements of all the desired system variables are available, and we seek to use these measurements in the initial period of time to deduce the full set of desired variables for the subsequent time, for which we assume that measurements of only a limited subset of the desired variables are possible. Our method utilizes a machine learning technique, called *reservoir computing* (see Ref. 6). This technique employs an input/output neural network with randomly generated parameters, and uses linear regression to choose "output weights" that fit the raw network output to a set of "training data." We use the data from the initial period of full measurement as the training data. Then we continue to input the subsequent partial set of continually measured variables, and regard the weighted network output as the estimated current values of the variables that are no longer measured. (We emphasize that the goal we address is the inference of unmeasured state variables rather than their prediction.)

The main result of this paper is that this kind of "reservoir observer," subject to certain limitations, can be extremely effective. In what follows we first describe a specific illustrative implementation and review relevant reservoir computing concepts. We then discuss applications to three examples of chaotic systems that highlight the strength and limitations of our method: (1) the Rössler system,[7] for which we have done an intensive study of how results depend on design parameters of the reservoir observer; (2) the Lorenz system,[8] which we use to illustrate an instance of the issue of observability for our method; and (3) the Kuramoto–Sivashinsky partial differential equation,[9–11] which we use to illustrate the possible effectiveness of our method in cases of spatiotemporal chaos.

II. SETUP

We consider a dynamical system $d\boldsymbol{\phi}/dt = \mathbf{f}(\boldsymbol{\phi})$ together with a pair of $\boldsymbol{\phi}$-dependent, vector valued variables, $\mathbf{u} = \mathbf{h}_1(\boldsymbol{\phi}) \in \mathbb{R}^M$ and $s = \mathbf{h}_2(\boldsymbol{\phi}) \in \mathbb{R}^P$. We are interested in the situation in which $\mathbf{u}$ and $\mathbf{s}$ can both be measured over a specific period, $[0, T]$, but that only $\mathbf{u}$ can be measured from that time forward; we seek a method for using the continued knowledge of $\mathbf{u}$ to determine an estimate of $\mathbf{s}$ as a function of time when direct measurement of $\mathbf{s}$ is not available, $t > T$. In contrast with most of the engineering literature devoted to problems of this kind, we do not assume knowledge of $\mathbf{f}$ but rather seek to infer the necessary information from the trajectories recorded on the interval $[0, T]$.

For this purpose we use "reservoir computing,"[6] which has previously been advocated for application to many tasks (e.g., prediction of time series, pattern recognition, etc.). There are many variations in implementation; in this paper we adopt the reservoir technique proposed by Jaeger and Haas.[12] The reservoir computer has three components (Fig. 1), a linear input layer with M input nodes (one for each component of $\mathbf{u}$), a recurrent, nonlinear reservoir network with N dynamical reservoir nodes whose state vector is $\mathbf{r} \in \mathbb{R}^N$, and a linear output layer with P output nodes, as shown in Fig. 1. For the specific reservoir computing implementation we use in this paper, the reservoir dynamics is defined as

$$\mathbf{r}(t + \Delta t) = (1 - \alpha)\mathbf{r}(t) + \alpha\tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t) + \xi 1), \tag{1}$$

where we assume the time step to be small $\Delta t \ll 1$, $\mathbf{A}$ is the (typically sparse) weighted adjacency matrix of the reservoir layer, and the M-dimensional input $\mathbf{u}(t)$ is fed in to the N reservoir nodes via a linear input weight matrix denoted by $\mathbf{W}_{in} \in \mathbb{R}^{N\times M}$. (In Secs. III A and III B, $M = 1$ and the input $\mathbf{u}$ is a scalar, while, in Sec. III C, $M \geq 1$.) The parameter $0 < \alpha \leq 1$ is a "leakage" rate[13] that makes the reservoir evolve more slowly as $\alpha \to 0$. We also use a bias term $\xi\mathbf{1}$, where $\mathbf{1}$ denotes a vector of ones and ξ is a scalar constant. The notation $\tanh(\cdot)$ with a vector argument is defined as the vector whose components are the hyperbolic tangents of the corresponding components of the argument vector. The output, which is a P-dimensional vector, is taken to be a linear function of the reservoir state

$$\hat{\mathbf{s}}(t) = \mathbf{W}_{out}\mathbf{r}(t) + \mathbf{c}. \tag{2}$$

As compared with other artificial neural network approaches, the advantage of reservoir computing is that training is made computationally feasible for relatively large N, since only the output weights $\mathbf{W}_{out}$ and the vector $\mathbf{c}$ are adjusted by the training process. (The input weight matrix $\mathbf{W}_{in}$ and the reservoir adjacency matrix $\mathbf{A}$ are initially randomly drawn and then fixed.) A key point is that the reservoir layer serves as an active medium driven by inputs $\mathbf{u}(t)$ where each reservoir node has a different nonlinear response to its inputs, so that for $N \gg 1$ we can hope that almost a wide variety of desired outputs can be approximated by a linear combination of the N-dimensional reservoir nodal response states.

In addition to the parameters Δt, α, and ξ in Eq. (1), and the reservoir size N, the reservoir dynamics depend on the parameters D, ρ, and σ, which govern the random generation of $\mathbf{A}$ and $\mathbf{W}_{in}$ as follows. The adjacency matrix $\mathbf{A}$ is built from a sparse random Erdős–Rényi matrix in which the fraction of nonzero matrix elements is D/N, so that the average degree of a reservoir node is D. The values of non-zero elements are randomly drawn independently from a uniform distribution between -1 and 1. We then uniformly rescale all the elements of $\mathbf{A}$ (i.e., multiply $\mathbf{A}$ by a positive scalar) so that the largest value of the magnitudes of its eigenvalues becomes ρ, which we refer to as the "spectral radius" of $\mathbf{A}$. For the input layer, the i-th of the M input signals is connected to N/M reservoir nodes with connection weights in the i-th column of $\mathbf{W}_{in}$. Each reservoir node receives input from exactly one input signal. The non-zero elements of $\mathbf{W}_{in}$ are randomly chosen from a uniform distribution in $[-\sigma, \sigma]$. (We emphasize that this choice of structure for $\mathbf{W}_{in}$ is fairly arbitrary and that other choices could reasonably be employed, e.g., every node could receive a mixture of several inputs.)

For the convenience of comparing the reservoir performances, we preprocess all the components of $\mathbf{u}(t)$ and $\mathbf{s}(t)$

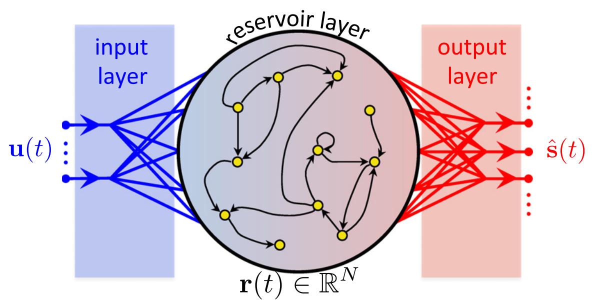


FIG. 1. A reservoir computer consisting of three parts, an input layer, a reservoir layer with state $\mathbf{r}(t)$, and an output layer. For $t > T$, the input to the system is $\mathbf{u}(t)$ and our goal is that the output $\hat{\mathbf{s}}(t)$ is a good approximation to the unmeasured quantity $\mathbf{s}(t)$.

so that they have zero mean and unit variance. Starting from a random initial state $\mathbf{r}(-\tau)$, the reservoir evolves following Eq. (1) with input $\mathbf{u}(t)$. Here τ is a transient time, chosen large enough to make the reservoir state essentially independent of its initial state by time $t=0$. We then record the $K=T/\Delta t$ reservoir states for $0<t\leq T$

$$\{\mathbf{r}(\Delta t), \mathbf{r}(2\Delta t), \ldots, \mathbf{r}(T)\}, \tag{3}$$

and the concurrent measurements of the state variables that are unmeasured for $t>T$

$$\{\mathbf{s}(\Delta t), \mathbf{s}(2\Delta t), \ldots, \mathbf{s}(T)\}. \tag{4}$$

We then train the network by choosing the output layer quantities $\mathbf{W}_{out}$ and $\mathbf{c}$ by choosing them so that the reservoir output approximates the measurement for $0<t\leq T$. We do this by minimizing the following quadratic form with respect to $\mathbf{W}_{out}$ and $\mathbf{c}$

$$\left\{\sum_{k=1}^{K} \|\mathbf{W}_{out}\mathbf{r}(k\Delta t) + \mathbf{c} - \mathbf{s}(k\Delta t)\|^2\right\} + \beta\left[\mathrm{Tr}(\mathbf{W}_{out}\mathbf{W}_{out}^T)\right], \tag{5}$$

where $\|\mathbf{q}\|^2 = \mathbf{q}^T\mathbf{q}$ for $\mathbf{q}$ a vector. The second term of Eq. (5), $\beta[\mathrm{Tr}(\mathbf{W}_{out}\mathbf{W}_{out}^T)]$, is a regularization term included to avoid overfitting $\mathbf{W}_{out}$, where $\beta>0$ (typically a small number) is the "ridge regression parameter."

If the training is successful, the readout of the reservoir output should yield a good approximation (denoted $\hat{\mathbf{s}}(t)$) to the desired unmeasured quantity $\mathbf{s}(t)$ for $t>T$. Referring to Eq. (2)

$$\hat{\mathbf{s}}(t) = \mathbf{W}_{out}^*\mathbf{r}(t) + \mathbf{c}^*, \tag{6}$$

where $\mathbf{W}_{out}^*$ and $\mathbf{c}^*$ denote the solutions for the minimizers of Eq. (5)

$$\mathbf{W}_{out}^* = \delta\mathbf{S}\delta\mathbf{R}^T(\delta\mathbf{R}\delta\mathbf{R}^T + \beta\mathbf{I})^{-1}, \tag{7}$$

$$\mathbf{c}^* = -\left[\mathbf{W}_{out}^*\bar{\mathbf{r}} - \bar{\mathbf{s}}\right], \tag{8}$$

$$\bar{\mathbf{r}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{r}(k\Delta t), \quad \bar{\mathbf{s}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{s}(k\Delta t), \tag{9}$$

where $\mathbf{I}$ is the $N\times N$ identity matrix, $\delta\mathbf{R}$ (respectively, $\delta\mathbf{S}$) is the matrix whose kth column is $\mathbf{r}(k\Delta t) - \bar{\mathbf{r}}$ (respectively, $\mathbf{s}(k\Delta t) - \bar{\mathbf{r}}$).

We remark that Eq. (1) represents a special choice for the form of the reservoir that is convenient for our purposes. More generally, Eq. (1) can be expressed as

$$\mathbf{r}(t+\Delta t) = \mathbf{g}(\mathbf{r}(t), \mathbf{u}(t)), \tag{10}$$

and other forms of $\mathbf{g}$, different from the choice Eq. (1), have been employed for reservoir methods designed to implement goals different from the observer goal that we address here. For example, experimental reservoir implementations have been reported where the dynamics Eq. (10) were from an optical network of semiconductor lasers,[14] a delay system with a single nonlinear node,[15] a field-programmable gate array (FPGA),[16] phase-delay electro-optic devices,[17] and even a bucket of water,[18] among others. Such choices might also work for a reservoir-based observer and may offer advantages such as the potential for huge increase in speed.[17] The main requirement on the observer dynamics Eq. (10) seems to be that it is sufficiently complex and that the dimension of the reservoir state vector $\mathbf{r}$ is sufficiently large that the output Eq. (2) can be made to approximate the desired time series (for our goal, $\mathbf{s}(t)$) by adjustment of $\mathbf{W}_{out}$ and $\mathbf{c}$.

III. EXAMPLES

In this section, we illustrate the reservoir-based observer method of Sec. II using three chaotic dynamical systems: the Rössler system, the Lorenz system, and the Kuramoto–Sivashinsky equations.

A. Rössler system

We now test the observing method given in Sec. II on a chaotic Rössler system

$$dx/dt = -y - z, \tag{11}$$

$$dy/dt = x + ay, \tag{12}$$

$$dz/dt = b + z(x-c), \tag{13}$$

where $a=0.5$, $b=2.0$, and $c=4.0$. We choose one of the coordinates of the system as the "continually available" measurement $\mathbf{u}(t)$ defined in Sec. II (i.e., $M=1$). The other two coordinates are taken to be the components of $\mathbf{s}(t)$, knowledge of which is only available for the training phase $0\leq t\leq T$.

We find that a trained reservoir computing network with input from any one of these three coordinates, x, y, and z, can generate output that accurately agrees with the two remaining coordinates. As shown in Fig. 2, a reservoir computer whose preprocessed input is

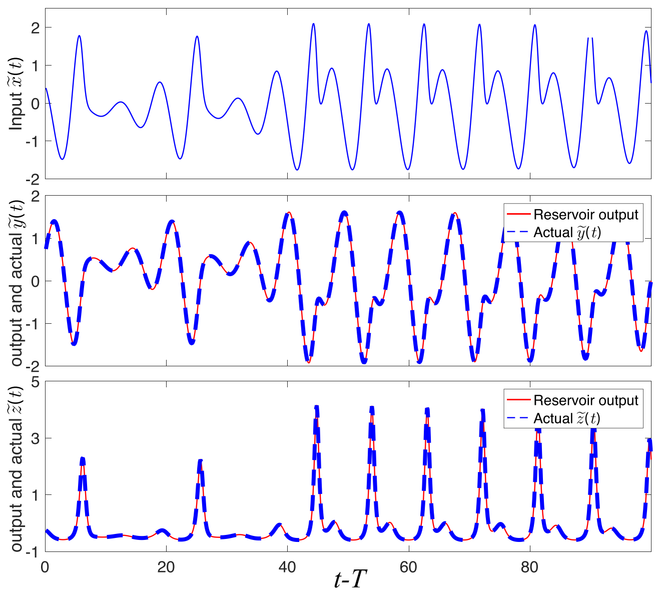


FIG. 2. For the Rössler system Eqs. (11)–(13) with input $\tilde{x}(t)$, the reservoir computer very accurately generates $\tilde{y}(t)$ and $\tilde{z}(t)$. The reservoir parameters we use here are given in Eq. (15).

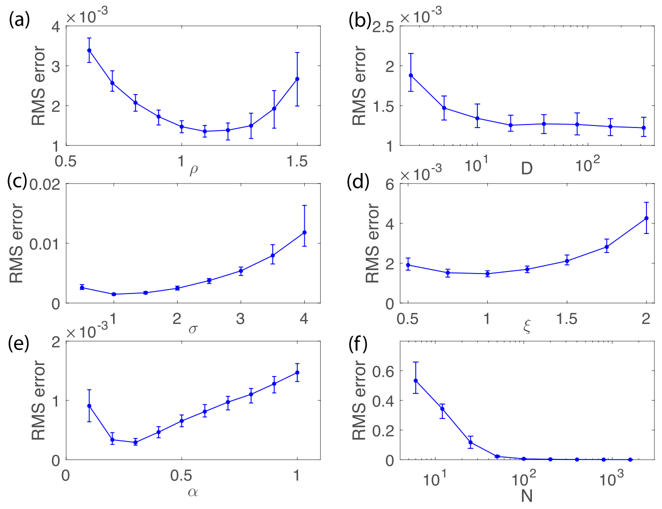


FIG. 3. The RMS error in the inference of $\tilde{y}(t)$ from the measurement of $\tilde{x}(t)$ versus one of the parameters (ρ, D, σ, ξ, α, or N), with the other parameters held fixed as given in Eq. (15). For the data in (f) with the RMS error on a logarithmic scale, see the curve in Fig. 4(a) labeled "without Eq. (16)."

$$\tilde{x}(t) = [x(t) - \langle x(t)\rangle]/\langle[x(t) - \langle x(t)\rangle]^2\rangle^{1/2} \qquad (14)$$

can accurately output the similarly defined versions ($\tilde{y}(t)$, $\tilde{z}(t)$) of ($y(t)$, $z(t)$) for $t > T$, i.e., $\mathbf{u}(t) = \tilde{x}(t)$, $\mathbf{s}(t) = [\tilde{y}(t), \tilde{z}(t)]^T$. (In Eq. (14) the angle brackets denote time average.)

For the results in Fig. 2 we utilize the following set of reservoir parameters (defined in Sec. II):

$$\begin{aligned} \text{number of reservoir nodes}: &\quad N = 400,\\ \text{spectral radius}: &\quad \rho = 1.0,\\ \text{average degree}: &\quad D = 20,\\ \text{scale of input weights}: &\quad \sigma = 1.0,\\ \text{bias constant}: &\quad \xi = 1.0,\\ \text{leakage rate}: &\quad \alpha = 1.0,\\ \text{time interval}: &\quad \Delta t = 0.1,\\ \text{length of training phase}: &\quad T = 260. \end{aligned} \qquad (15)$$

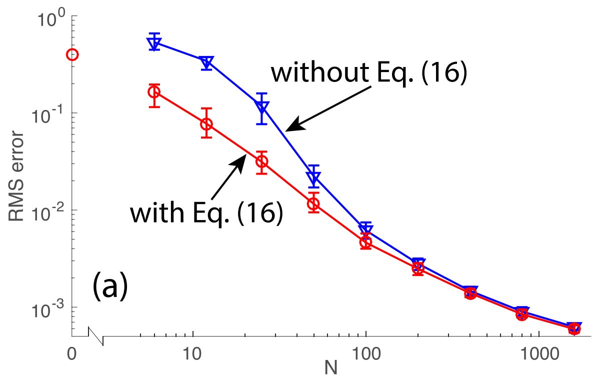
For this set of parameters the difference between the inferred (thin line in Figs. 2(b) and 2(c)) and actual (dashed line for Figs. 2(b) and 2(c)) values of $\tilde{y}(t)$ and $\tilde{z}(t)$ is very small.

We examined how the root mean square (RMS) error of the inferred values varies with the reservoir parameters. Results of this type are given in Figs. 3(a)–3(f) where we plot the RMS error over 500 time units in the inference of $\tilde{y}(t)$ from the measurement of $\tilde{x}(t)$ versus one of the parameters (ρ, D, σ, ξ, α, or N), with the other parameters held fixed as given in Eq. (15). Each plotted point in these figures is the median of 100 trials using the same signals $\tilde{x}(t)$ and $\tilde{y}(t)$, with each trial using an independent random realization of the reservoir system, and the error bars indicate the range in the performance of the observer from the second to the third quartile. Except for plots (b) and (f), the same 100 random choices are used for each value of the parameter. For example, Fig. 3(a) is a plot of the RMS error versus ρ. There is a wide range for which the error is small ($\lesssim 10^{-2}$), thus demonstrating that good observer performance is robust to significant changes in ρ. The plots in Figs. 3(a)–3(f) show that good performance is robust to changes in the other parameters, as well.

Motivated by the frequent consideration[19–21] of time delayed measurements for the analysis of experimental data from chaotic systems, we have also tested incorporation of time delayed versions of the input into the output by adding the term

$$\sum_{k=0}^{K} d_k \mathbf{u}(t - k\Delta t), \qquad (16)$$

into the reservoir output Eq. (2) and correspondingly also into the expression given by Eq. (5). We then minimize the expression in Eq. (5) over $\mathbf{W}_{out}$, $\mathbf{c}$, and $\mathbf{d} = (d_1, d_2, \dots, d_K)^T$ to obtain our approximation $\hat{\mathbf{s}}$ for $\mathbf{s}$. An illustrative result is shown in Fig. 4(a), where we plot the median RMS error in $\tilde{y}$ inferred from $\tilde{x}(t)$ as a function of N for the two cases where Eq. (16) is included ($K = 20$ for Fig. 4(a)) and not included. We find that including delays does not significantly improve performance for $N > 100$, but does make some improvement when the number of reservoir nodes N is small and the error is relatively larger. For example, if one requires the error to be less than about 3×10^{-2}, then one can use as few as $N \sim 20$ reservoir nodes when the delays are included, while, without delays, $N \sim 50$ is required. We note that for $N = 0$ (no reservoir), the procedure reduces to a standard linear



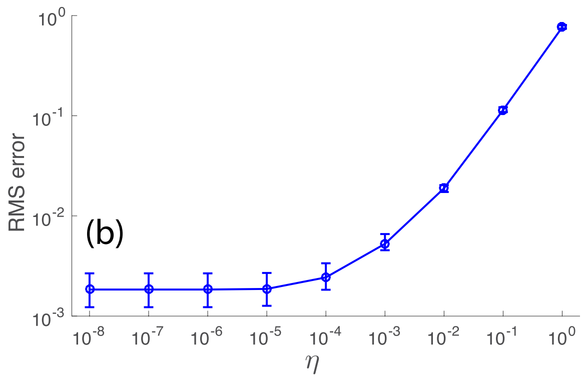


FIG. 4. (a) The RMS error of the inference of $\tilde{y}(t)$ from $\tilde{x}(t)$ versus the size of the reservoir N with Eq. (16) and $K = 20$ and without Eq. (16). Aside from N, all the parameter values used in this figure are as given in Eq. (15). (b) The effect of observational noise η in both $\tilde{x}(t)$ and $\tilde{y}(t)$ on the RMS error. The reservoir parameters for this plot are as given in Eq. (15). For both figures, the plotted points indicate the median error over 100 random realizations of the reservoir and the error bars indicate the range of the second and the third RMS error quartiles.

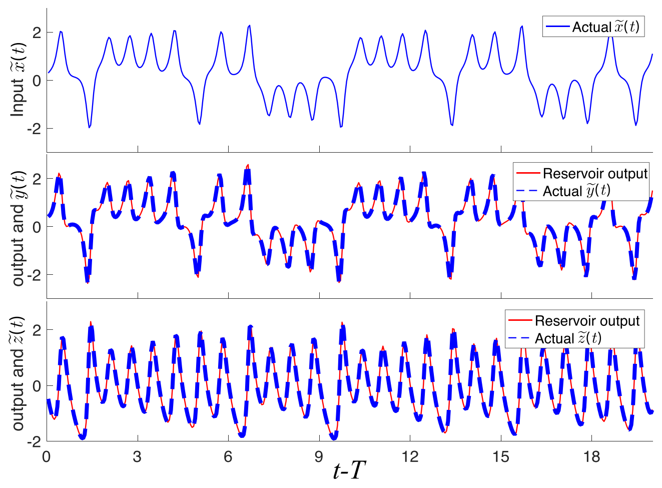


FIG. 5. For the Lorenz system Eq. (17) with input $\tilde{x}(t)$, the reservoir computer very accurately generates $\tilde{y}(t)$ and $\tilde{z}(t)$. The reservoir parameters we use here and in Figs. 6 and 7 are the same as those given in Eq. (15) except that $\Delta t = 0.05$.

autoregressive approximation of **s**. We also tested other values of K and found that the quantitative results were very much the same as for $K = 20$ (plotted as open circles in Fig. 4(a)) as long as $K > 5$.

Figure 4(b) shows the effect of observational noise in the input $\tilde{x}(t)$ and the training data $\tilde{y}(t)$, where, at every time step $\Delta t = 0.1$, observational noise was simulated by adding an independent random number with uniform distribution between $-\eta$ and $+\eta$ to each "measurement." We see that the noise has little influence on the error for $\eta \leq 10^{-4}$.

B. Lorenz system

In this section we test our method on the Lorenz system. This example provides insight into how the issue of "observability" manifests itself for our method.

The Lorenz system is described by the equations

$$\begin{aligned} dx/dt &= -ax + ay, \\ dy/dt &= bx - y - xz, \\ dz/dt &= cz + xy, \end{aligned} \tag{17}$$

where $a = 10$, $b = 28$, and $c = 8/3$. As in Sec. III A, the time series $x(t)$, $y(t)$, and $z(t)$ are available during the training phase, $0 < t < T$. Also, as in Sec. III A, time series are pre-processed as in Eq. (14) to give $\tilde{x}(t), \tilde{y}(t)$, and $\tilde{z}(t)$.

First we consider the case where $\mathbf{u}(t) = \tilde{x}(t)$. After the training phase is over, the reservoir can only access the series $\tilde{x}(t)$ and is able to infer $\tilde{y}(t)$ and $\tilde{z}(t)$ accurately as shown in Fig. 5. Clearly, there is no observability problem for $\mathbf{u}(t) = \tilde{x}(t)$.

However, if $\mathbf{u}(t) = \tilde{z}(t)$, then observability does not apply, i.e., it is not possible, even in principle, to infer $\tilde{x}(t)$ from $\tilde{z}(t)$. To see this, we note that the Lorenz system is invariant under the transformation $(x, y, z) \to (-x, -y, z)$. Because of this symmetry, any given time series measurement $z_0(t)$ can correspond to either one of the two possible state time series on the attractor $(x_0(t), y_0(t), z_0(t))$ and $(-x_0(t), -y_0(t), z_0(t))$. This results in the system's "non-observability" when only the z coordinate is measured. Figure 6(a) shows that with $\tilde{z}(t)$ as input the reservoir output for $\tilde{x}(t)$ (solid line) completely fails to follow the actual $\tilde{x}(t)$ trajectory (dashed line). A similar result applies for $\tilde{y}(t)$.

However, since the ambiguity in the state lies purely in the sign of the x and y variables, it should be possible, in principle, to infer the state variables (x^2, y^2) unambiguously given measurements of the z variable. We show in Fig. 6(b) that this is indeed the case.

Another way to demonstrate the interplay, in this example, between the symmetry of the system and observability is to modify the Lorenz system slightly in a way that breaks the symmetry. The modified set of Lorenz equations is given by Eq. (17), but with an extra term $x(t)$ added to the right hand side of the third equation (i.e., the equation for dz/dt). With this change, a visualization of the attractor (not shown) in (x, y, z) state space shows that the attractor is slightly perturbed from the picture obtained for the original Lorenz system, now with the size and orientations of the two lobes (wings of Lorenz's "butterfly attractor") being slightly different from each other. As shown in Fig. 7, the reservoir is now able to infer the x and y state variables from measurement of the z variable.

It is interesting to note that, while, as shown in Fig. 5, inference of z when either x or y is given can work well, and inference of z completely fails when $\xi = 0$. The reason for this is interesting. Since the hyperbolic tangent in Eq. (1) is an odd function, when $\xi = 0$, the composite system

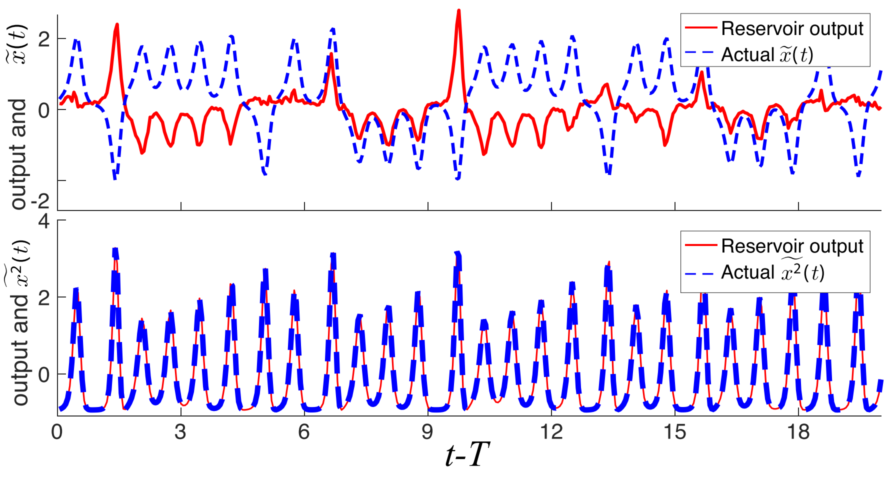


FIG. 6. (a) A reservoir computer with input $\tilde{z}(t)$ from the Lorenz system Eq. (17) cannot generate $\tilde{x}(t)$. (b) However, it can accurately generate $\widetilde{x^2}(t) = [x^2(t) - \langle x^2(t)\rangle]/\langle [x^2(t) - \langle x^2(t)\rangle]^2\rangle^{1/2}$, and, as in Eq. (14), the angle brackets denote time averages.

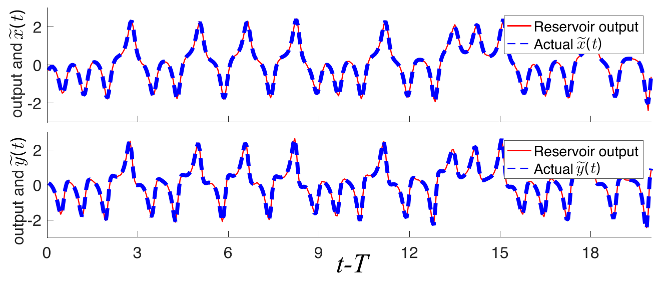


FIG. 7. A reservoir computer with input $\tilde{z}(t)$ from the modified Lorenz system with its symmetry broken. In this case, the reservoir accurately generates $\tilde{x}(t)$ and $\tilde{y}(t)$.

consisting of reservoir plus the Lorenz equations has the overall symmetry $(x, y, z, \mathbf{r}) \to (-x, -y, z, -\mathbf{r})$. Thus, we see that, although (x, y) and $(-x, -y)$ may correspond to the same value of z, this is incompatible with the linear output dependence hypothesized in Eq. (2), since $\mathbf{r} \to -\mathbf{r}$ would necessarily change the output estimate of z. However, if $\xi \neq 0$ the symmetry of the composite Lorenz-reservoir system is broken, and inference of z becomes potentially possible. (Even with $\xi = 0$, another alternative, which we have found to work, is to introduce terms quadratic in $\mathbf{r}$ into Eq. (2).)

We remark that the symmetry mechanism we focused on in this subsection is meant only as one particular example of the very many ways that nonobservability can arise.

C. Kuramoto–Sivashinsky equations

We now investigate the possibility of using the reservoir-based observation method of Sec. II to infer estimates of the state of a spatiotemporally chaotic system from spatially sparse measurements without the use of a model.

For this purpose, we test our model-free observation method on simulated data from the Kuramoto–Sivashinsky equation[22] for the scalar variable $y(x, t)$

$$y_t = -yy_x - y_{xx} - y_{xxxx}. \tag{18}$$

We impose spatially periodic boundary conditions, i.e., $y(x+L, t) = y(x, t)$ on a domain $\{x \in (0, L)\}$ of size $L = 22$ and integrate Eq. (18) from a randomly chosen initial condition. The integration was performed on an evenly spaced grid of size $Q = 65$. The simulated data consist of Q time series with a time step $\Delta t = 0.25$ units. We denote this set of time series at the Q grid points by the vector $\mathbf{y}(t) = (y_1(t), y_2(t), \ldots, y_Q(t))^T$ with $y_i(t) = y(i\Delta x), \Delta x = L/(Q-1)$. Let $\mathbf{u}(t) = (u_1(t), u_2(t), \ldots u_M(t))^T$ be a vector containing M out of the Q time series in $\mathbf{y}(t)$. In terms of the variables $\mathbf{s}(t)$ and $\mathbf{u}(t)$ introduced in Sec. II, the vector $\mathbf{u}(t)$ represents spatially sparse measurements performed at evenly spaced points on the grid. We denote the rest of the time series by $\mathbf{s}(t)$. We will vary the number of measurements M in the interval $1 \leq M \leq 8$. We assume that we have access to the full set of state measurements $\mathbf{y}(t)$ for the "training period," $0 \leq t \leq T$. We further assume that after the training period, i.e., for $t > T$, the observer can only access the partial set of system state variables $\mathbf{u}(t)$. The goal is to infer the set of variables $\mathbf{s}(t)$ from the partial measurements $\mathbf{u}(t)$ for $t > T$.

The reservoir observer setup is described in Sec. II with the identification $Q = M + P$. For the results in this subsection, we use the following set of reservoir parameters:

$$\begin{aligned} N &= 3000, & \rho &= 0.9, \\ D &= 60, & \sigma &= 0.5, \\ \xi &= 0.0, & \alpha &= 0.3, \\ \Delta t &= 0.25, & T &= 15\,000. \end{aligned} \tag{19}$$

To test the quality of the inference for $t \geq T$, we compare the inferred data from the reservoir $\hat{\mathbf{s}}(t)$ with the data series $\mathbf{s}(t)$ obtained from integrating Eq. (18). We calculate the correlation between the real and inferred data, defined by

$$C = \frac{\left(\sum_{i,t}(s_i(t) - \langle s\rangle)(\hat{s}_i(t) - \langle \hat{s}\rangle)\right)}{\sqrt{\left(\sum_{i,t}(s_i(t) - \langle s\rangle)^2\right)\left(\sum_{i,t}(\hat{s}_i(t) - \langle \hat{s}\rangle)^2\right)}} \tag{20}$$

and the spatially averaged RMS error, defined by

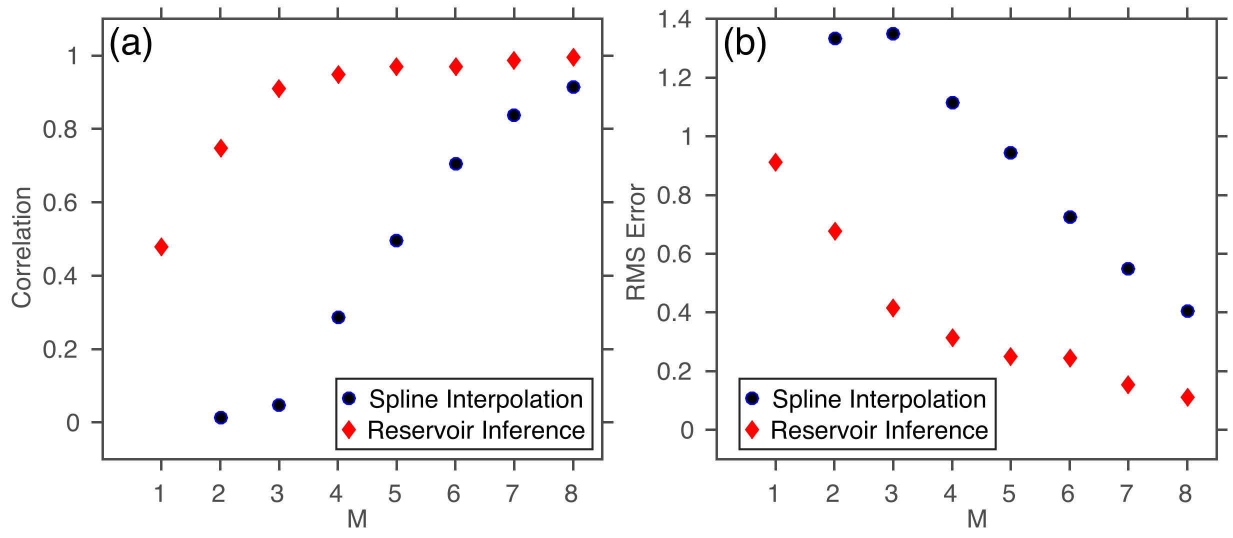


FIG. 8. (a) Correlation between observer inferred data and the actual data for the Kuramoto–Sivashinsky state and (b) RMS error in the inference of the Kuramoto–Sivashinsky state variables versus the number of measured variables M. Standard deviations are smaller than the size of the symbols.

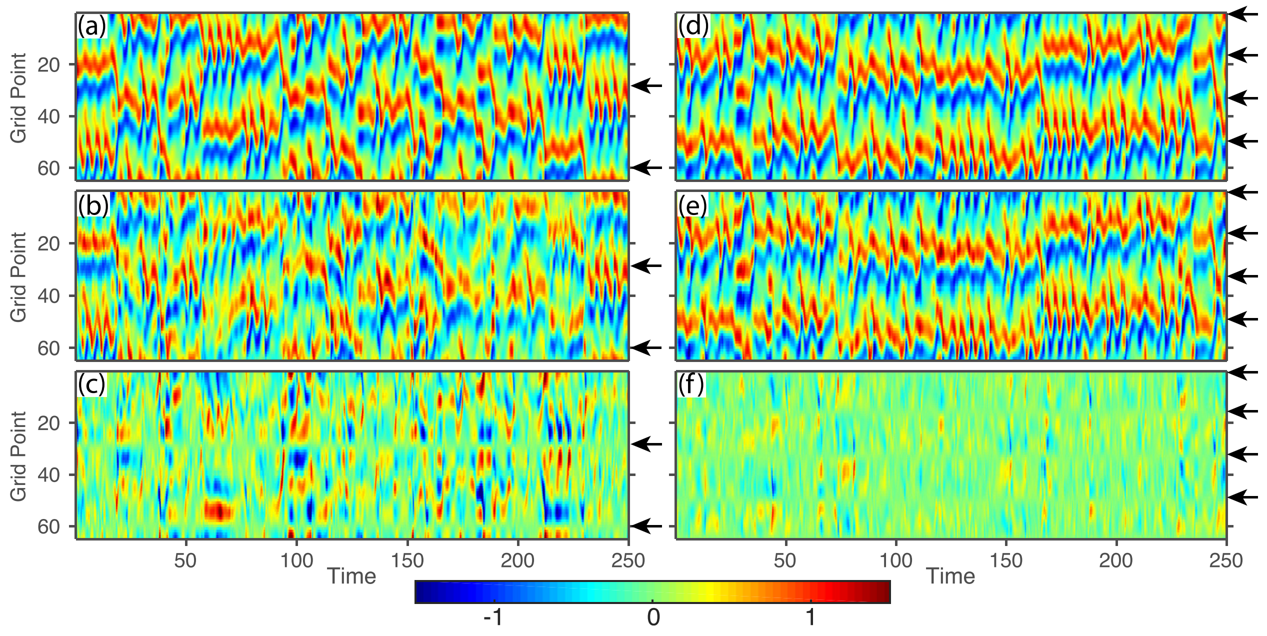


FIG. 9. Results from two simulations, (a)–(c) and (d)–(f), where (a)–(c) have $M = 2$ inputs, and (d)–(f) have $M = 4$ inputs, whose locations are indicated by the black arrows. The top panels, (a) and (d), show the actual state evolution $y(x, t)$ of the Kuramoto–Sivashinsky system. The middle panels, (b) and (e), show the evolution of the state inferred by the reservoir observer from the measurements. The bottom panels, (c) and (f), show the difference between the inferred data and the actual data.

$$R = \sqrt{\frac{\sum_{i,t}[s_i(t) - \hat{s}_i(t)]^2}{\sum_{i,t}[s_i(t)]^2}}. \tag{21}$$

Since the performance of the reservoir may depend on the particular random instance of the reservoir network that is used, we report the mean RMS error and the correlation between the inferred and actual data obtained from 20 observer trials each performing the inference task on the same data using an independent random realization of the reservoir observer setup. As a baseline comparison, we also report the RMS error and correlation coefficient values for an inference of the unmeasured variables obtained by cubic spline interpolation from the measured variables. Figure 8(a) shows the correlation between the reservoir inference and the actual data as we vary the number of measured variables. The correlation coefficient for the reservoir observer inference is compared with the corresponding value for the cubic spline interpolation scheme. Figure 8(b) shows the RMS error obtained by the reservoir setup as we vary the number of measured variables. These figures demonstrate that, as expected, spline interpolation yields good results at high measurement densities, but that, at lower measurement densities, where spline interpolation yields poor results, the reservoir observer can continue to function well. Figure 9 shows results for $M = 2$ inputs (Figs. 9(a)–9(c)) and $M = 4$ inputs (Figs. 9(d)–9(f)) for the actual data (Figs. 9(a) and 9(d)) and the reservoir inference (Figs. 9(b) and 9(e)), together with the difference between the inferred and the actual data (Figs. 9(c) and 9(f)). Even for $M = 2$ inputs the error is quite low in most of the plot (Figs. 9(c)) with bursts of error occurring intermittently in time and space.

IV. CONCLUSIONS

In this paper we investigate the application of reservoir computing to infer unmeasured state variables of a chaotic dynamical system from a limited set of continually measured state variables for situations in which a mathematical model of the dynamical system is unavailable or is insufficiently accurate. Our main results are as follows.

(1) Extremely accurate results can be robustly obtained over a wide range of reservoir parameters as shown in Sec. III A for the Rössler system.
(2) The issue of "observability" and the limitation it implies for achieving our goal was investigated in an example (Sec. III B).
(3) Use of reservoir computers for inference of spatiotemporally chaotic states from spatially sparse data was proposed and validated by application to an example (Sec. III C).

ACKNOWLEDGMENTS

We thank D. Gauthier and A. Hartemink for their useful discussions. This work was supported by the Army Research Office under Grant No. W911NF-12-1-0101.

[1]K. Ogata and Y. Yang, *Modern Control Engineering* (Prentice-Hall, Upper Saddle River, NJ, 1970).
[2]R. Hermann and A. J. Krener, IEEE Trans. Autom. Control **22**, 728 (1977).
[3]P. So, E. Ott, and W. Dayawansa, Phys. Rev. E **49**, 2650 (1994).
[4]J. C. Quinn, P. H. Bryant, D. R. Creveling, S. R. Klein, and H. D. Abarbanel, Phys. Rev. E **80**, 016201 (2009).
[5]J. Schumann-Bischoff, S. Luther, and U. Parlitz, Phys. Rev. E **94**, 032221 (2016).
[6]M. Lukoševičius and H. Jaeger, Comput. Sci. Rev. **3**, 127 (2009).
[7]O. E. Rössler, Phys. Lett. A **57**, 397 (1976).
[8]E. N. Lorenz, J. Atmos. Sci. **20**, 130 (1963).

[9]B. I. Cohen, J. Krommes, W. Tang, and M. Rosenbluth, Nucl. Fusion **16**, 971 (1976).
[10]Y. Kuramoto and T. Tsuzuki, Prog. Theor. Phys. **55**, 356 (1976).
[11]G. Sivashinsky, Acta Astronaut. **4**, 1177 (1977).
[12]H. Jaeger and H. Haas, Science **304**, 78 (2004).
[13]H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, Neural Networks **20**, 335 (2007).
[14]D. Brunner, S. Reitzenstein, and I. Fischer, in *IEEE International Conference on Rebooting Computing (ICRC)* (IEEE, 2016), pp. 1–2.
[15]L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Nat. Commun. **2**, 468 (2011).
[16]N. D. Haynes, M. C. Soriano, D. P. Rosin, I. Fischer, and D. J. Gauthier, Phys. Rev. E **91**, 020801 (2015).
[17]L. Larger, A. Baylón-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, and M. Jacquot, Phys. Rev. X **7**, 011015 (2017).
[18]C. Fernando and S. Sojakka, in *European Conference on Artificial Life* (Springer, 2003), pp. 588–597.
[19]F. Takens, in *Dynamical Systems and Turbulence, Warwick 1980* (Springer, 1981), pp. 366–381.
[20]A. Brandstater and H. L. Swinney, Phys. Rev. A **35**, 2207 (1987).
[21]T. Sauer, J. A. Yorke, and M. Casdagli, J. Stat. Phys. **65**, 579 (1991).
[22]J. M. Hyman and B. Nicolaenko, Phys. D: Nonlinear Phenom. **18**, 113 (1986).