Chaos
An Interdisciplinary Journal of Nonlinear Science

RESEARCH ARTICLE | OCTOBER 30 2023

Detecting disturbances in network-coupled dynamical systems with machine learning

Special Collection: Data-Driven Models and Analysis of Complex Systems

Per Sebastian Skardal ; Juan G. Restrepo

Check for updates

Chaos 33, 103137 (2023)
https://doi.org/10.1063/5.0169237

View Online Export Citation

Detecting disturbances in network-coupled dynamical systems with machine learning

Cite as: Chaos **33**, 103137 (2023); doi: 10.1063/5.0169237
Submitted: 24 July 2023 · Accepted: 5 October 2023 ·
Published Online: 30 October 2023

Per Sebastian Skardal[1,a] and Juan G. Restrepo[2]

AFFILIATIONS

[1] Department of Mathematics, Trinity College, Hartford, Connecticut 06106, USA
[2] Department of Applied Mathematics, University of Colorado Boulder, Boulder, Colorado 80309, USA

Note: This paper is part of the Focus Issue: Data-Driven Models and Analysis of Complex Systems.
[a] **Author to whom correspondence should be addressed:** persebastian.skardal@trincoll.edu

ABSTRACT

Identifying disturbances in network-coupled dynamical systems without knowledge of the disturbances or underlying dynamics is a problem with a wide range of applications. For example, one might want to know which nodes in the network are being disturbed and identify the type of disturbance. Here, we present a model-free method based on machine learning to identify such unknown disturbances based only on prior observations of the system when forced by a known training function. We find that this method is able to identify the locations and properties of many different types of unknown disturbances using a variety of known forcing functions. We illustrate our results with both linear and nonlinear disturbances using food web and neuronal activity models. Finally, we discuss how to scale our method to large networks.

Despite a wide range of potential applications, identifying disturbances made to network-coupled dynamical systems is a difficult problem due to the complex nature of interactions between different units. Even a disturbance made to a single node can be challenging to localize due to the propagation of behavior through the network. Here, we present a model-free method using machine learning techniques, specifically reservoir computing, to identify disturbances to network-coupled dynamical systems. This method assumes no knowledge of the underlying dynamics or the disturbance itself. All that is needed is sufficient observations of the system under the influence of a known forcing function. We show using examples from ecology and neuroscience that the method robustly identifies the disturbances made to the system for a relatively simple set of forcing functions used for training. Moreover, we show that the method is scalable to large networks using a pseudo-parallelization architecture.

I. INTRODUCTION

Machine learning techniques have proven to be extremely useful for data-driven modeling and prediction of complex systems.[1–4] In many of these applications, a machine learning system is trained to learn and replicate the dynamics of a nonlinear system from noisy or partially observed data, and then the machine learning system is used, for example, to forecast the dynamics,[5–8] to estimate Lyapunov exponents[9] or unstable periodic orbits,[10] to infer network coupling,[11] or to predict extreme events[12] and crises in non-stationary dynamical systems.[13,14]

Machine learning techniques can also be used without the need to replicate the intrinsic dynamics of the system. For example, Ref. 15 uses reservoir computers, a particular class of machine learning systems suited to modeling time-dependent systems, to learn the response of dynamical systems to stimuli and then design data-driven control algorithms. Reservoir computing has also been used in other engineering tasks,[39] such as recovering signals in noisy communications,[16] detecting wind and rain events,[17] and identifying physiological properties of biosynthetic materials.[18] In Ref. 19, we recently proposed a similar scheme to identify and suppress unknown disturbances to dynamical systems. Detecting disturbances to nonlinear dynamical systems is a crucial problem with a wide range of applications, e.g., in engineering, and, in particular, in power grid networks,[20–26] ecology,[27,28] fluid dynamics,[29,30] and climate change.[31] In this paper, we extend our results to the identification of disturbances in network-coupled dynamical systems. Network-coupled systems present a particular challenge in identifying disturbances, as the ripple of an external force complicates the inference of both the location and nature of the disturbance.[25,32,33]

By extending the method presented in Ref. 19 to network-coupled dynamical systems, we are able to robustly identify which nodes are disturbed and the time course of the disturbances. We illustrate our approach with two examples: a food web and a system of excitatory and inhibitory neuron populations. The dynamics in these examples include both stationary and oscillatory dynamics and linear and nonlinear disturbances. We also address the issue of the scalability of our approach to large networks. We believe our approach could be useful in applications where identifying the location of disturbances in networked dynamical systems is important, such as, for example, power grid systems.

The remainder of this paper is organized as follows. In Sec. II, we present the problem statement and describe the setup of the reservoir computer. In Sec. III, we present our first example: linear disturbances to a food web. In Sec. IV, we present our second example: nonlinear disturbances to a network of excitatory and inhibitory neuron populations. In Sec. V, we explore the scalability of our approach to larger networks using an ensemble of node-specific reservoirs. In Sec. VI, we conclude with a discussion of our results.

II. PROBLEM STATEMENT AND SETUP OF THE RESERVOIR COMPUTER

We consider here systems of N network-coupled dynamical systems whose states $\boldsymbol{x}_i(t) \in \mathbb{R}^D$ for $i = 1, \dots, N$ are governed by a system of differential equations of the form

$$\dot{\boldsymbol{x}}_i = \boldsymbol{F}_i(\boldsymbol{x}_i, \boldsymbol{X}, \boldsymbol{g}_i(t)), \tag{1}$$

where $\boldsymbol{X}(t) = [\boldsymbol{x}_1(t)^T, \dots, \boldsymbol{x}_N(t)^T]^T \in \mathbb{R}^{ND}$ is the collection of all state vectors $\boldsymbol{x}_i(t)$ organized in a single vector, $\boldsymbol{g}_i(t) \in \mathbb{R}^D$ is the disturbance to the state at node i, and the vector field $\boldsymbol{F}_i : \mathbb{R}^D \times \mathbb{R}^{ND} \times \mathbb{R}^D \to \mathbb{R}^D$ incorporates the local intrinsic dynamics of state i, the effect of interactions with other states via $\boldsymbol{X}$ and some underlying network structure, and the local disturbance $\boldsymbol{g}_i$. In particular, we write the vector field $\boldsymbol{F}_i$ such that $\boldsymbol{F}_i(\boldsymbol{x}_i, \boldsymbol{X}, \boldsymbol{0}) = \boldsymbol{F}_{i,\text{intrinsic}}(\boldsymbol{x}_i, \boldsymbol{X})$ gives the intrinsic, i.e., undisturbed, system dynamics. In general, the disturbances $\boldsymbol{g}_i(t)$ may be linear, in which case the dynamics may be written as $\boldsymbol{F}_i(\boldsymbol{x}_i, \boldsymbol{X}, \boldsymbol{g}_i(t)) = \boldsymbol{F}_{i,\text{intrinsic}}(\boldsymbol{x}_i, \boldsymbol{X}) + \boldsymbol{g}_i(t)$, or nonlinear, in which case no such notational simplification can be made. Our goal is to develop a method by which the disturbances can be accurately identified in such a way that we can infer (i) precisely which element(s) of the system are disturbed and (ii) what the disturbance functions are. We assume that the system can be forced with a set of known *training forcing functions* $\boldsymbol{h}_i(t) \in \mathbb{R}^D$ for $i = 1, \dots, N$, as

$$\dot{\hat{\boldsymbol{x}}}_i = \boldsymbol{F}_i(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{X}}, \boldsymbol{h}_i(t)), \tag{2}$$

and each $\hat{\boldsymbol{x}}_i(t)$ can be observed for a sufficiently long time. (From now on, a hat will indicate quantities during the training phase). Then, a machine learning system is trained to approximate each $\boldsymbol{h}_i(t)$ given all $\hat{\boldsymbol{x}}_i(t)$. The trained machine learning system is then used to infer $\boldsymbol{g}_i(t)$ from observations of $\boldsymbol{x}_i(t)$ obtained from Eq. (1). Note that no knowledge of the intrinsic dynamics $\boldsymbol{F}_{i,\text{intrinsic}}$ or the disturbance functions $\boldsymbol{g}_i$ is required.

As our machine-learning system implementation, we will use reservoir computers, a class of machine learning systems particularly well-suited for time-dependent problems.[4] We assume that the training system given in Eq. (2) [i.e., with known forcing functions $\boldsymbol{h}_i(t)$] is first run on a training interval $[-\hat{T}, 0]$, and a time series of the observed state vector $\{\hat{\boldsymbol{X}}(-\hat{T}), \hat{\boldsymbol{X}}(-\hat{T} + \Delta t), \dots, \hat{\boldsymbol{X}}(0)\}$ is collected. At each time step, these variables are fed into the reservoir, a high-dimensional dynamical system with internal variables $\boldsymbol{r} \in \mathbb{R}^M$, where M is the size of the reservoir. Here, we implement the reservoir as

$$\boldsymbol{r}(t + \Delta t) = \tanh[A\boldsymbol{r}(t) + W_{\text{in}}\hat{\boldsymbol{X}}(t) + 1], \tag{3}$$

where the $M \times M$ matrix A is a sparse matrix representing the internal structure of the reservoir and the $M \times ND$ matrix W_{in} is a fixed input matrix. At each time, the reservoir output $\boldsymbol{u}$ is constructed from the internal states as $\boldsymbol{U} = W_{\text{out}}\boldsymbol{r}$, where $\boldsymbol{U}(-n\Delta t) = [\boldsymbol{u}_1(t)^T, \dots, \boldsymbol{u}_N(t)^T]^T \in \mathbb{R}^{ND}$ organizes the vectors $\boldsymbol{u}_i(t)$ into a single vector and the $ND \times M$ output matrix W_{out} is chosen so that the reservoir outputs $\mathbf{u}_i(t)$ are a good approximation to each known training forcing function $\boldsymbol{h}_i(t)$. Thus, the output matrix W_{out} is the only component of the reservoir that needs to be trained, and this can be done by minimizing the cost function

$$\sum_{n=0}^{\hat{T}/\Delta t} \|\boldsymbol{H}(-n\Delta t) - \boldsymbol{U}(-n\Delta t)\|^2 + \lambda \text{Tr}\left(W_{\text{out}} W_{\text{out}}^{\text{T}}\right), \tag{4}$$

via a ridge regression procedure, where a small constant $\lambda \geq 0$ is used to prevent over-fitting and $\boldsymbol{H}(t) = [\boldsymbol{h}_1(t)^T, \dots, \boldsymbol{h}_N(t)^T]^T \in \mathbb{R}^{ND}$ organizes the vectors $\boldsymbol{h}_i(t)$ into a single vector. This procedure trains the reservoir to identify the forcing functions $\boldsymbol{h}_i(t)$ given the observations of $\hat{\boldsymbol{x}}_i(t)$, and subsequently, the reservoir can be presented with a time series of the observed variables $\boldsymbol{x}_i(t)$ from Eq. (1) in an interval $[0, T]$ and evolved as

$$\boldsymbol{r}(t + \Delta t) = \tanh[A\boldsymbol{r}(t) + W_{\text{in}}\boldsymbol{X}(t) + 1]. \tag{5}$$

If the method works as intended, the reservoir output $\boldsymbol{U}(t) = W_{\text{out}}\boldsymbol{r}$ on the interval $[0, T]$ will be a good approximation to the unknown disturbance, $\boldsymbol{u}_i \approx \boldsymbol{g}_i$, where $\boldsymbol{g}_i(t) = \boldsymbol{0}$ for nodes i that are not directly disturbed. We define the vector $\boldsymbol{G}(t) = [\boldsymbol{g}_1(t)^T, \dots, \boldsymbol{g}_N(t)^T]^T \in \mathbb{R}^{ND}$ to organize the disturbance vectors $\boldsymbol{g}_i(t)$ into a single large vector. Reference 19 illustrates how the method performs when the intrinsic dynamics consists of a single Lorenz chaotic system.

In Fig. 1, we illustrate the setup of our reservoir computer. First, during the training phase, i.e., in the interval $[-\hat{T}, 0]$, the known forcing functions $\boldsymbol{h}_i(t)$ [organized in $\boldsymbol{H}(t)$] are fed into the reservoir (top left), while the observed system states $\hat{\boldsymbol{x}}_i(t)$ [organized in $\hat{\boldsymbol{X}}(t)$] force the reservoir dynamics (center), and the output matrix W_{out} is trained to recover $\boldsymbol{U}(t) \approx \boldsymbol{H}(t)$ (top right). When training is complete, the unknown disturbances $\boldsymbol{g}_i(t)$ [organized in $\boldsymbol{G}(t)$] are fed into the reservoir (bottom left), while the new observed system states $\boldsymbol{x}_i(t)$ force the reservoir dynamics (center), and $\boldsymbol{U}(t) \approx \boldsymbol{G}(t)$ is recovered (bottom right).

Throughout this paper, unless otherwise specified, the reservoir matrix A is a random matrix of size $M = 1000$ with entries A_{ij} uniformly distributed in $[-0.5, 0.5]$ with probability $6/M$ and otherwise, $A_{ij} = 0$, and then rescaled to set its spectral radius μ to 1.2. (We note that while this choice of μ does not guarantee the echo state property, we find favorable results that are comparable

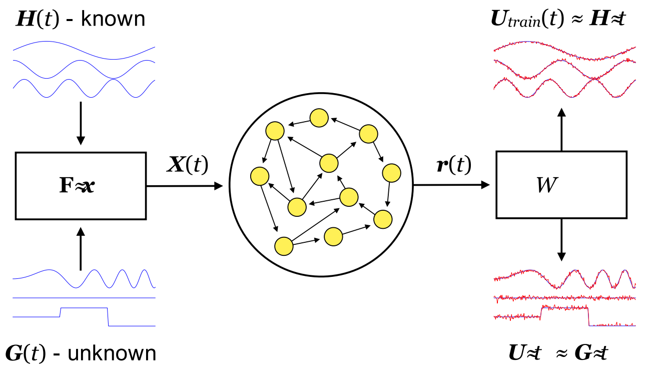

FIG. 1. Reservoir computer architecture. An illustration of the setup of our reservoir computer. During the training phase, the known forcing functions $\boldsymbol{h}_i(t)$ [organized in $\boldsymbol{H}(t)$] force the dynamics (top left), which produces the states $\hat{\boldsymbol{x}}_i(t)$ [organized in $\hat{\boldsymbol{X}}(t)$] and subsequently feed the reservoir (center), whose variables $\boldsymbol{r}(t)$ are used to train the output matrix W_{out} to recover $\boldsymbol{U}(t) \approx \boldsymbol{H}(t)$ (top right). Next, the system is disturbed with the unknown functions $\boldsymbol{g}_i(t)$ [organized in $\boldsymbol{G}(t)$] (bottom left), which produces $\boldsymbol{X}(t)$, then feeds the reservoir (center), at which point the new variables $\boldsymbol{r}(t)$ are used to recover $\boldsymbol{U}(t) \approx \boldsymbol{G}(t)$ (bottom right).

across a range of μ values both less than and greater than one; see Fig. 10.) The input matrix W_{in} is a random matrix where each entry is uniformly distributed in $[-0.01, 0.01]$. The ridge regression regularization constant is $\lambda = 10^{-6}$. Training times $\hat{T}$ and time steps Δt vary depending on the chosen dynamics and will be given below with each example. Moreover, in the majority of examples considered here, we will consider either periodic or quasiperiodic disturbances to a system that, in isolation, settles to a fixed point or a periodic system that is driven by a constant or slowly varying disturbance that does not break the periodicity of the dynamics. In both cases, largest Lyapunov exponents are zero. In one example (see Fig. 7), the forced dynamics fall to a fixed point, yielding a negative largest Lyapunov exponent.

III. EXAMPLE WITH LINEAR FORCING: DISTURBANCES TO A FOOD WEB

As a first example, we consider the generalized Lotka–Volterra model, composed of N interacting species whose populations (which we will also refer to as *biomasses*) are denoted as $x_i(t) \geq 0$.[40] Note that the state of each species is a scalar; therefore, each $x_i(t)$, $h_i(t)$, and $g_i(t)$ is of dimension $D = 1$. Each biomass evolves according to

$$\dot{x}_i = x_i\left(e_i - \frac{x_i}{K_i} + \sum_{j=1}^{N} P_{ij}x_j\right) + g_i(t), \tag{6}$$

where e_i represents the linear growth rate of species i, e_iK_i is the carrying capacity of species i, and $g_i(t)$ is the disturbance made to species i. Note that in this example, disturbances are linear; i.e., each vector field $\boldsymbol{F}_i$ may be written as $\boldsymbol{F}_i(x_i, \boldsymbol{X}, g_i(t)) = \boldsymbol{F}_{i,\text{intrinsic}}(x_i, \boldsymbol{X}) + g_i(t)$. The $N \times N$ matrix P encodes the interactions between species such that $P_{ij} > 0$ indicates that the presence of species j is favorable for species i, $P_{ij} < 0$ indicates that the presence of species j is unfavorable for species i, and $P_{ij} = 0$ if no direct interaction exists. Here, we consider an $N = 8$ species example whose interactions are those of predator–prey dynamics; i.e., for every interacting pair (i, j), P_{ij} and P_{ji} are of opposite signs. In Fig. 2, we illustrate this network, indicating all parameters and organizing species hierarchically with predators at the top and prey at the bottom. The dynamics of the undisturbed system, i.e., with $g_i(t) = 0$ for all $i = 1, \ldots, N$, are plotted in Fig. 3(a) as the biomasses come to a stable fixed point representing coexistence. For this system, we use a time step of $\Delta t = 0.005$, updating the dynamics using Heun's method starting from a randomly chosen initial condition.

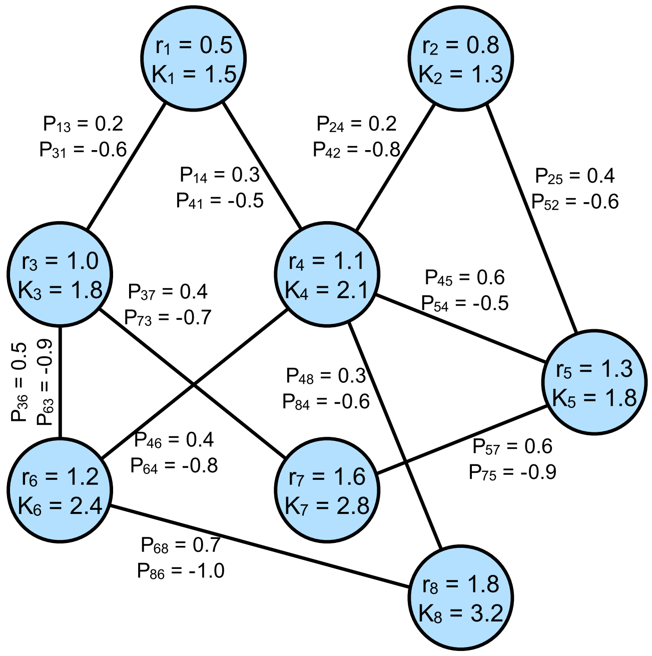

FIG. 2. Lotka–Volterra model. An illustration of the $N = 8$ species Lotka–Volterra model used here with the system given in Eq. (6). All parameters (linear growth rates r_i, carrying capacities K_i, and interaction strengths A_{ij}) are given. Species are hierarchically organized with predators at the top and prey at the bottom, respectively.

We now consider identifying and recovering unknown disturbances to this system using the reservoir computer architecture described above. For the training phase, we first consider a training interval of length $\hat{T} = 100$ using forcing functions $h_i(t)$ composed of sinusoids with randomly chosen frequencies, specifically $h_i(t) = 0.8\sin(\omega_i t)$, where each ω_i is randomly distributed in the interval $[1, 9]$. We then consider disturbances made only to species $i = 3$ and 5 of the form $g_3(t) = 0.3\sin(2t) + 0.3\sin(\pi t)$ and $g_5(t) = \sin(2\pi\sin(t/2))$, and all other $g_i(t) = 0$. Results are plotted in Fig. 3 with the biomasses $x_i(t)$ under the known forcing and the unknown disturbance plotted in panels (b) and (c), respectively, and the forcing functions and disturbance functions plotted in panels (d) and (e), respectively. The actual training and disturbance functions $h_i(t)$ and $g_i(t)$ are plotted in solid blue, while the recovered functions are plotted in dashed red. As we can see in Fig. 3(e), despite the

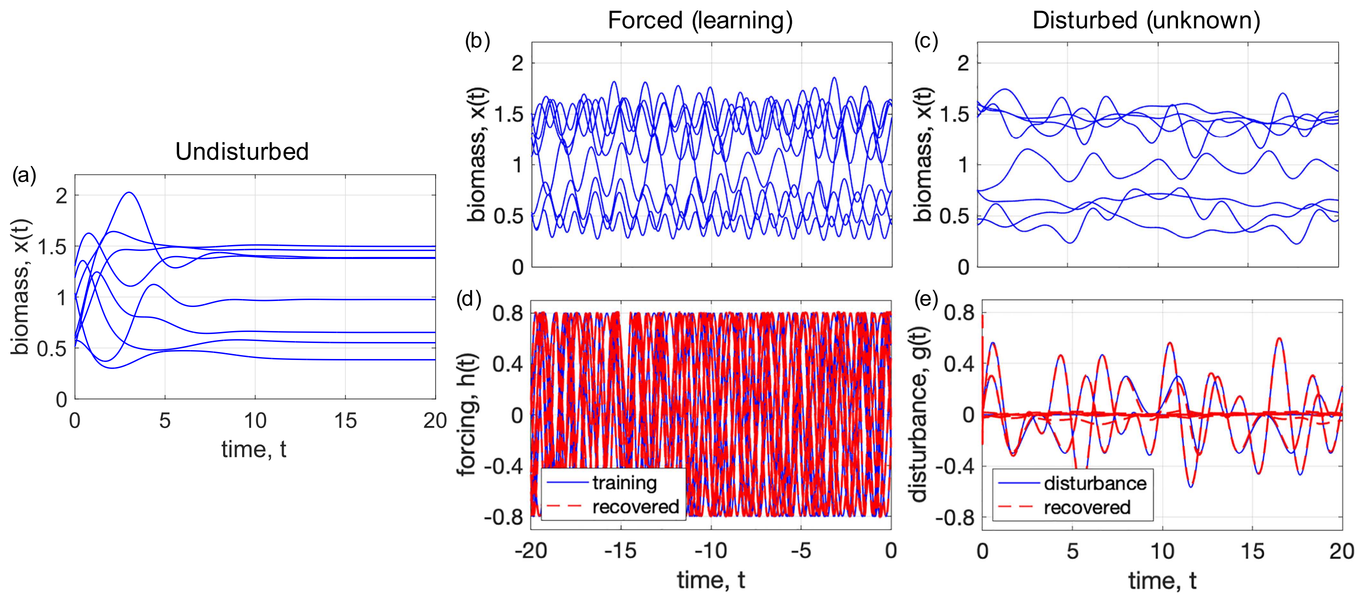

FIG. 3. Lotka–Volterra dynamics and disturbance recovery. (a) Intrinsic dynamics of the undisturbed Lotka–Volterra system. (b) and (c) Biomasses $x_i(t)$ of the Lotka–Volterra system under sinusoidal forcing $[h_i(t) = 0.8\sin(\omega_i t)]$ and pseudo-sinusoidal disturbance $[g_3(t) = 0.3\sin(2t) + 0.3\sin(\pi t)$ and $g_5(t) = \sin(2\pi\sin(t/2))]$, respectively. (d) and (e) Forcing and disturbance functions $h_i(t)$ and $g_i(t)$, respectively, with the actual and recovered functions plotted in solid blue and dashed red, respectively.

reservoir computer not knowing either the intrinsic system dynamics or the disturbances $g_i(t)$, the locations and nature of the disturbances are accurately recovered.

Before proceeding to a different example, a few remarks regarding the training forcing are in order. First, we used here a combination of sinusoids with mismatched (random) frequencies. This mismatch in frequencies allows (assuming a long enough training length $\hat{T}$) the forcing functions $h_i(t)$ to robustly explore all directions of forcing, which is necessary for accurately recovering disturbances.[15,19] As we will see next, the form of the forcing given as sinusoids here can be generalized as long as a robust range of forcing is considered. Second, we also make note that in principle, some non-zero forcing should be used at each node in general. If a specific node is unforced during training, i.e., $h_i(t) = 0$, then the method cannot recover any non-zero disturbance at that node.

In a second scenario, we consider a training interval of length $\hat{T} = 200$ using forcing functions $h_i(t)$ composed of random step functions. Each $h_i(t)$ is partitioned into 20 intervals (each interval lasts 10 time units) with values chosen uniformly at random in $[-0.01, 0.19]$. Disturbances are then made again only to species $i = 3$ and 5 in the form of Heaviside functions that transition from 0 to -0.1 and 0.15, respectively, at $t = 6.667$. Results are plotted in Fig. 4 with the biomasses $x_i(t)$ under the known forcing and the unknown disturbance plotted in panels (a) and (b), respectively, and the forcing functions and the disturbance functions plotted in panels (c) and (d), respectively. Again, the actual training and disturbances functions $h_i(t)$ and $g_i(t)$ are plotted in solid blue, while the recovered functions are plotted in dashed red, showing a robust identification of the disturbances.

IV. EXAMPLE WITH NONLINEAR FORCING: WILSON–COWAN NEURONS

We now turn to a second example and consider a network of interacting Wilson–Cowan neuron populations.[41] At its most minimal, the Wilson–Cowan model consists of one population of excitatory neurons and another population of inhibitory neurons whose aggregate activities are nonlinearly coupled; i.e., $D = 2$. Here, we consider a network consisting of $N = 4$ pairs of populations (i.e., yielding a system of total size $ND = 8$) where each pair consists of one excitatory population and one inhibitory population. The aggregate activity of each excitatory and inhibitory population is given by E_i and I_i with $i = 1, \dots, N$ that evolve according to

$$\tau \dot{E}_i = -E_i + S\left(w_{EE}E_i - w_{EI}I_i + P_i - w_{\text{net}}\sum_{j=1}^{N} B_{ij}I_j + g_i(t)\right), \quad (7)$$

$$\tau \dot{I}_i = -I_i + S\left(w_{IE}E_i - w_{II}I_i\right), \quad (8)$$

where

$$S(x) = \frac{Kx^2}{\sigma^2 + x^2} \quad (9)$$

is a sigmoidal response function. Note that each population has a linear response component in addition to the nonlinear response. Within the response function, both the excitatory and inhibitory populations play their namesake roles (coupling parameters w_{EE}, w_{EI}, w_{IE}, and w_{II} are all positive), while the excitatory populations are fed a baseline stimulus P_i. Moreover, each pair is coupled through the adjacency matrix B and network coupling parameter w_{net} in

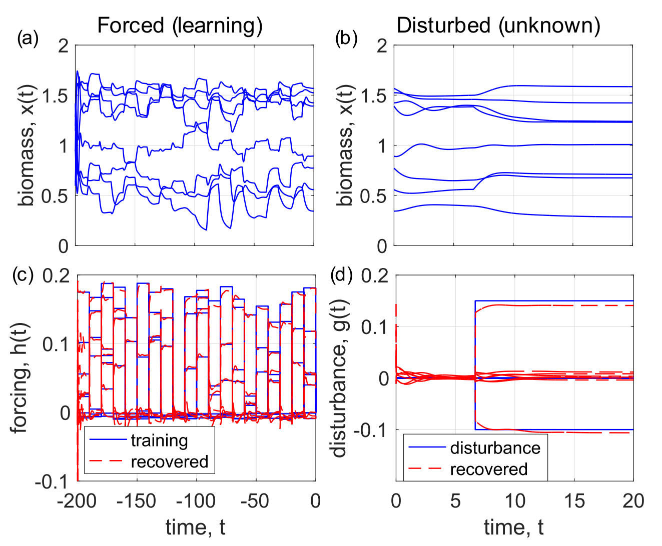

FIG. 4. Lotka–Volterra disturbance recovery: a second example. (a) and (b) Biomasses $x_i(t)$ of the Lotka–Volterra system under random step forcing and Heaviside disturbance, respectively. (c) and (d) Forcing and disturbance functions $h_i(t)$ and $g_i(t)$, respectively, with the actual and recovered functions are plotted in solid blue and dashed red, respectively.

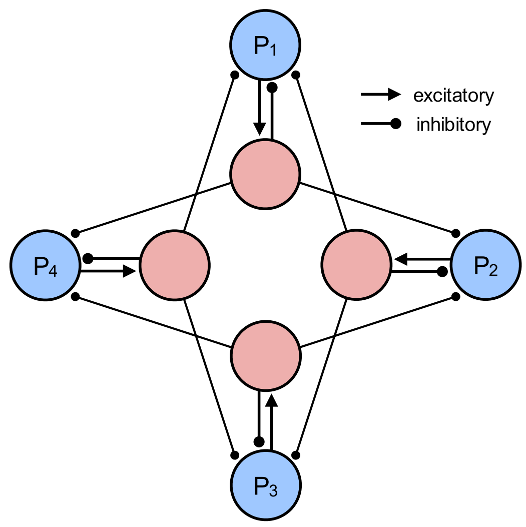

FIG. 5. Wilson–Cowan model. Illustration of the $n = 4$ ($N = 8$) population Wilson–Cowan model used here with the system given in Eqs. (9) and (10). Inhibitory and excitatory populations are organized on the inside and outside, respectively.

such a way that inhibitory populations affect not only their local excitatory counterpart, but also other excitatory populations according to the adjacency matrix. Finally, note that the disturbances in this system are nonlinear; i.e., functions $g_i(t)$ appear within the sigmoidal response functions, leading to nonlinear disturbances. Our system of $N = 4$ pairs of populations is illustrated in Fig. 5. Parameters are given by $\tau = 10$, $w_{EE} = 6.4$, $w_{EI} = 6.0$, $w_{IE} = 4.8$, $w_{II} = 1.2$, $w_{\text{net}} = 0.5$, $K = 1$, and $\sigma = 1$, as well as two sets of baseline stimuli P_i to give rise to two different dynamical regimes. Specifically, we consider a larger set of stimuli, $\boldsymbol{P} = [3.22, 3.02, 3.18, 3.33]^T$, and a set of smaller stimuli, $\boldsymbol{P} = [1.52, 1.61, 1.57, 1.64]^T$, which give rise to stationary dynamics and oscillatory dynamics, respectively, as illustrated in Figs. 6(a) and 6(b). Here, we use a time step of $\Delta t = 0.2$. We also consider disturbances made only to the excitatory nodes, which can be interpreted as modifications of the external stimuli P_i. Thus, in the notation adopted in the problem description, each state vector is given by $\boldsymbol{x}_i(t) = [E_i(t), I_i(t)]^T$, and we consider forcing and disturbances of the form $\boldsymbol{h}_i(t) = [h_i(t), 0]^T$ and $\boldsymbol{g}_i(t) = [g_i(t), 0]^T$.

Beginning with the dynamics in the stationary regime (i.e., we use $\boldsymbol{P} = [3.22, 3.02, 3.18, 3.33]^T$), we consider a training interval of length $\hat{T} = 4000$ and, drawing from the previous example, use forcing functions $h_i(t)$ composed of random steps. Each $h_i(t)$ is partitioned into 12 intervals with values chosen uniformly at random in $[-0.4, 0.6]$. Disturbances are then made again only to populations $i = 1$ and 3 in the form of Heaviside functions that transition from 0 to 0.4 and -0.3, respectively, at $t = 150$ and are back to zero at time $t = 400$. Results are plotted in Fig. 7 with the population activity $E_i(t)$ (solid blue) and $I_i(t)$ (dashed red) under the known forcing and the unknown disturbance plotted in panels (a) and (b), respectively, and the forcing functions and the disturbance functions plotted in panels (c) and (d), respectively. Again, the actual training and disturbance functions $h_i(t)$ and $g_i(t)$ are plotted in solid blue, while the recovered functions are plotted in dashed red, showing a robust identification of the disturbances.

Moving onto the oscillatory regime (i.e., using $\boldsymbol{P} = [1.52, 1.61, 1.57, 1.64]^T$), we find more nuanced results. Maintaining the same training setup as above, we now consider disturbing a single excitatory population with a composed sigmoidal function, $g_1(t) = -\frac{1}{2}\frac{1}{1+\exp(-(t-150)/30)} + \frac{1}{1+\exp(-(t-300)/30)}$, which effectively first decreases and then increases the baseline stimulus. Our initial attempt at recovering this disturbance is convoluted by the oscillatory dynamics, as can be seen in the results plotted in

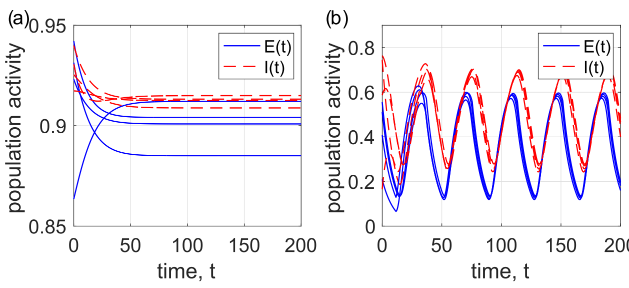

FIG. 6. Wilson–Cowan model: undisturbed dynamics. Population activities $E_i(t)$ and $I_i(t)$ of the undisturbed Wilson–Cowan system in the (a) stationary and (b) oscillatory regimes.

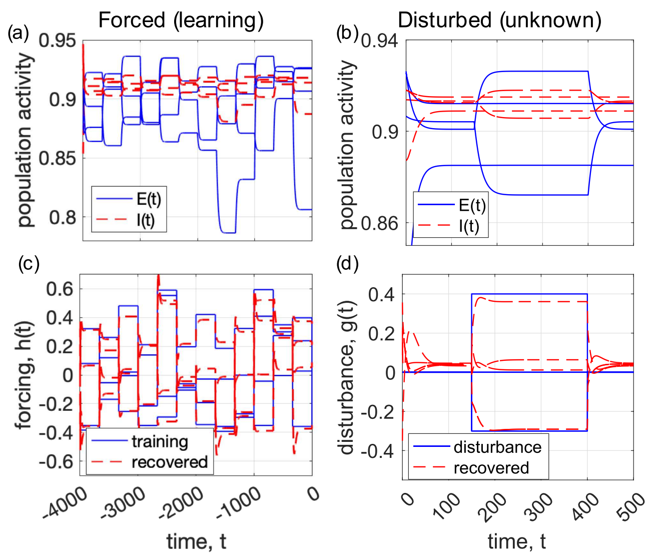

FIG. 7. Wilson–Cowan disturbance recovery: stationary regime. (a) and (b) Population activities $E_i(t)$ and $I_i(t)$ of the Wilson–Cowan system under random step forcing and Heaviside disturbance, respectively. (c) and (d) Forcing and disturbance functions $h_i(t)$ and $g_i(t)$, respectively, with the actual and recovered functions are plotted in solid blue and dashed red, respectively.

Figs. 8(a) and 8(d), where we plot the known forcing and the unknown disturbance, respectively, again with solid blue and dashed red denoting the actual and recovered signals. Note that the reservoir has trouble recovering the training functions $h_i(t)$, specifically averaging out the system's oscillations, which propagates into the recovery of the disturbances $g_i(t)$. Overall, the reservoir computer is able to produce a signal whose whole temporally localized mean is accurate, but losing precision due to strong oscillations. To address this issue, we adjust the reservoir dynamics to include a leak parameter $a_{\text{leak}} \in [0, 1]$; specifically, we consider dynamics of the form

$$\begin{aligned} \boldsymbol{r}(t+\Delta t) &= a_{\text{leak}}\boldsymbol{r}(t) \\ &\quad + (1-a_{\text{leak}})\tanh[A\boldsymbol{r}(t) + W_{\text{in}}\hat{\boldsymbol{x}}(t) + 1], \end{aligned} \tag{10}$$

where $a_{\text{leak}} = 0$ recovers the reservoir dynamics used previously [i.e., Eq. (3)] and with larger values essentially slowing down the dynamics. Using this adjustment of the reservoir dynamics, first plotting the forcing and disturbance results in Figs. 8(b) and 8(e) for $a_{\text{leak}} = 0.65$, then in Figs. 8(c) and 8(f) for $a_{\text{leak}} = 0.95$, quenches the oscillations recovered in both the training and disturbance phases, improving results. This improvement, however, comes at the cost of slowing down the reservoir dynamics and choosing a_{leak} too large can eventually degrade results, depending on the dynamics and forcing/disturbance.

V. SCALABILITY TO LARGE NETWORKS AND ERROR ANALYSIS

Before closing, we investigate further the scaling and error of the method presented in this paper. We begin by considering the case of larger networks than those presented above, and how this framework can be scaled to obtain accurate results, even for networks with many more units. For this purpose, we return to the Lotka–Volterra model used in Sec. III, which describes biomasses $x_i(t)$ for $i = 1, \ldots, N$ via Eq. (6). To generate such systems for arbitrary size N, we consider networks where each species has on average four interactions, a minimum of two interactions, and each non-zero interaction strength P_{ij} has an absolute value randomly and uniformly drawn from the interval $[2/N, 4/N]$ with all upper-triangular entries $(j > i)$ being positive and lower-triangular entries $(j < i)$ positive (negative) with probability 1/2 (1/2). Note that this suggests that roughly half of all interactions represent a predator–prey interaction, while the other half are mutualistic. In addition, each growth rate e_i is drawn randomly and uniformly from the interval $[2, 4]$, and each K_i is drawn randomly and uniformly from the interval $[1, 2]$. We also ensure that a strictly positive equilibrium of coexistence exists, with all species reaching equilibria at a value as large or larger than one; i.e., all fixed point values x_i^* satisfy $x_i^* \geq 1$.

Designing a reservoir computer that performs accurately as system size increases, however, is nontrivial. The predictive ability of traditional reservoir computer implementations quickly degrades as the network size grows[6] even as the size of the reservoir scales with the size of the system. While several methods have been recently explored and shown to improve the performance of reservoir computers,[34–38] here, we follow the architecture presented in Ref. 6. Specifically, we consider an alternative pseudo-parallel architecture constructed as follows. For a system that is comprised of N coupled dynamical units, we construct N individual reservoirs that each corresponds to one of the dynamical units in the system. Each of these individual reservoirs, indexed $i = 1, \ldots, N$, acts as described in Sec. II but (i) takes as an input the dynamics of node i via $\boldsymbol{x}_i(t)$ along with the dynamics of all the network neighbors of i via those $\boldsymbol{x}_j(t)$ for which j is connected to i and (ii) aims to recover only the disturbance $\boldsymbol{g}_i(t)$ [and in training the forcing $\boldsymbol{h}_i(t)$]. In short, reservoir i is itself driven by the dynamics of i and its network neighbors and is trained to recover the disturbance made to node i. Note that the influence of network neighbors makes this architecture not strictly parallelizable; therefore, we refer to it as the pseudo-parallel scheme. To ensure that computations involving each reservoir remain feasible, we choose each individual reservoir to be of a fixed size M (with an $M \times M$ adjacency matrix governing the internal dynamics of the reservoir), resulting in a total of N reservoirs of size N, i.e., NM dynamical reservoir units. For comparison, we also consider standard single reservoir architectures of size NM, i.e., the same number of dynamical reservoir units, in order to evaluate the performance of the pseudo-parallel scheme.

We now compare the results obtained using the standard and pseudo-parallelized architectures for Lotka–Volterra systems described above over a range of system sizes N with reservoir size parameter $M = 100$. For each case, we use a training interval of length $\hat{T} = 100$ with training functions $h_i(t) = 0.8\sin(\omega_i t)$, where each frequency ω_i is randomly and uniformly drawn from the interval $[1, 9]$. We then consider disturbances to 20% of nodes (the remaining 80% are undisturbed) with either the function $g_i(t) = u_1\sin(u_2 t) + u_1\sin(u_3 t)$ or $u_1\sin[u_4\sin(u_2 t)]$ (chosen with probabilities 1/2 and 1/2) where, for each node, u_1, u_2, u_3, and u_4 are randomly and uniformly drawn from the intervals

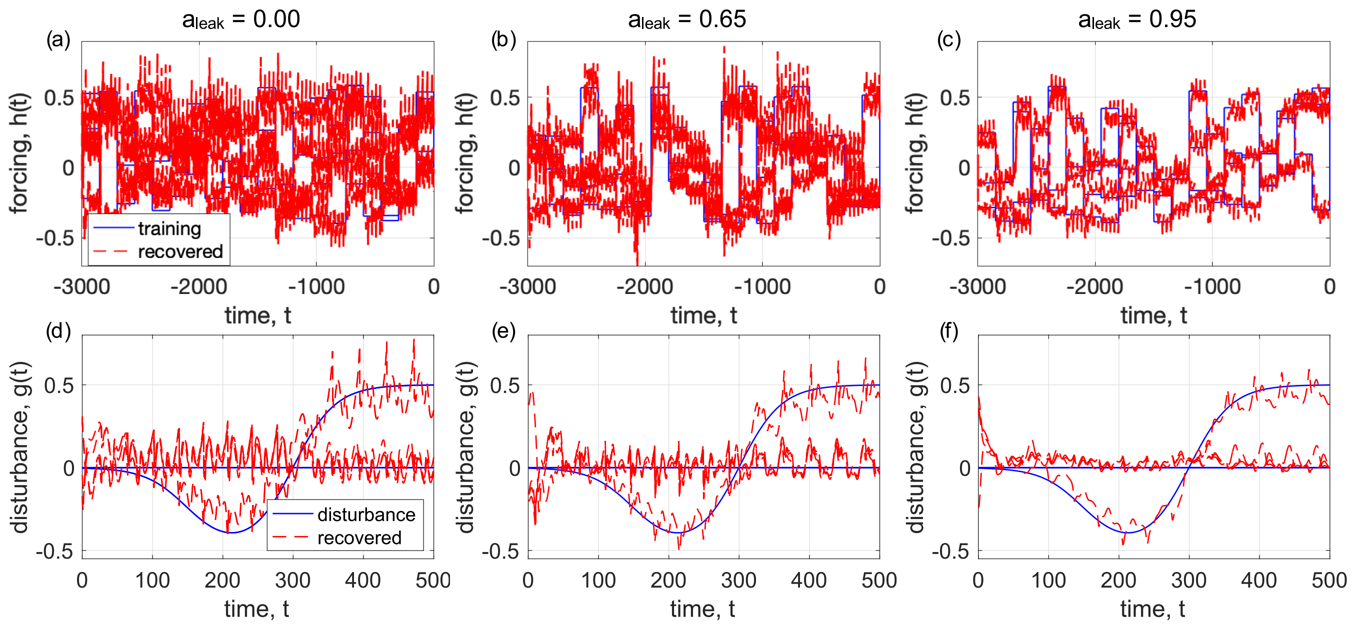

FIG. 8. Wilson–Cowan disturbance recovery: oscillatory regime. Using leak parameters $a_{\mathrm{leak}} = 0$, 0.65, and 0.95 the actual and recovered (solid blue and dashed red, respectively) forcing functions $h_i(t)$ plot in panels (a)–(c), along with corresponding actual and recovered disturbance functions $g_i(t)$ plotted in panels (d)–(f).

[0.2, 0.4], [1, 2], $[\pi/2, \pi]$, and $[\pi, 2\pi]$, respectively. We consider 20 realization of each system size N, plotting in Fig. 9 the mean squared error aggregated over all realizations for the standard and pseudo-parallel schemes in blue circles and red triangles, respectively. Results plotted in panels (a) and (b) represent errors taken from disturbed and undisturbed nodes, respectively. As system sizes increase from $N = 5$ to 120, we see that while the error in results obtained by using the standard reservoir scheme increases steadily when the pseudo-parallelized scheme is used, the error remains roughly constant. Thus, by implementing the pseudo-parallelized scheme, the overall framework for identifying disturbances in network-coupled dynamical systems scales and remains robust even as larger networks are considered, maintaining accurate results while providing significant computational savings for large N. Thus, the pseudo-parallel scheme is able to maintain an acceptable level of error since each single reservoir is tasked with recovering a single disturbance, contrasting with the standard scheme in which a single (albeit large) reservoir begins to fail as it is asked to recover more and more disturbances.

Next, we return to the smaller Lotka–Volterra model illustrated in Fig. 2 in an error analysis where the method presented here may be compared to some useful base cases. We start by varying the spectral radius μ of the adjacency matrix A inside the reservoir. Note that in the limit $\mu \to 0^+$, the matrix A vanishes, leading to reservoir dynamics that are solely driven by the input $\boldsymbol{X}(t)$. Increasing μ then increases the overall influence of the internal dynamics of the reservoir relative to the input from $\boldsymbol{X}(t)$. In Fig. 10, we plot the results obtained from varying μ between 0 and 2, plotting the mean squared error between the true and recovered disturbance. Each

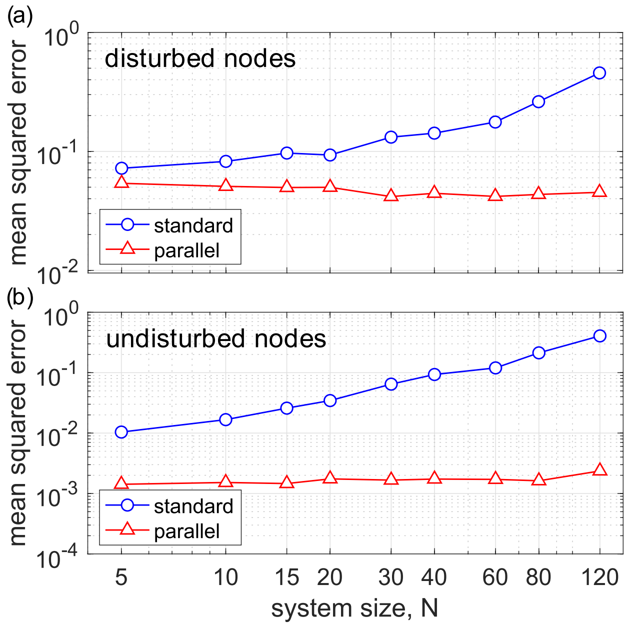

FIG. 9. Standard vs pseudo-parallel schemes: Error comparison. As a function of system size N, a comparison of the mean squared error for the standard (blue circles) and pseudo-parallel (red triangles) schemes that use, respectively, one large reservoir of size NM and N smaller reservoirs or size M, with $M = 100$. Results in panels (a) and (b) represent errors calculated from disturbed and undisturbed nodes, respectively.

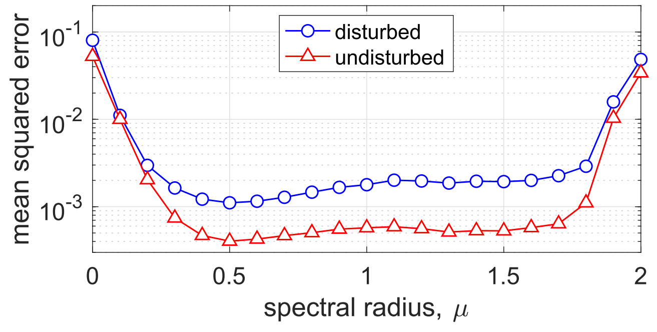

FIG. 10. Effect of a spectral radius: Error analysis. As a function of the spectral radius μ of the adjacency matrix A in the reservoir dynamics, the mean squared error between true and recovered disturbances for disturbed (blue circles) and undisturbed nodes (red triangles). Underlying dynamics are given by the Lotka–Volterra model illustrated in Fig. 2 with disturbances made to nodes 3 and 5 using $g_3(t) = u_1 \sin(u_2 t) + u_1 \sin(u_3 t)$ and $g_5(t) = u_1 \sin[u_4 \sin(u_2 t)]$ with u_1, u_2, u_3, and u_4 drawn randomly and uniformly from [0.2, 0.4], [1, 2], $[\pi/2, \pi]$, and $[\pi, 2\pi]$, respectively.

data point represents the average over 50 realizations, where at each realization, nodes 3 and 5 were disturbed with the functions $g_3(t) = u_1 \sin(u_2 t) + u_1 \sin(u_3 t)$ and $g_5(t) = u_1 \sin[u_4 \sin(u_2 t)]$, where, as above, u_1, u_2, u_3, and u_4 were randomly and uniformly drawn from the intervals $[0.2, 0.4]$, $[1, 2]$, $[\pi/2, \pi]$, and $[\pi, 2\pi]$. The error calculated for the disturbed and undisturbed nodes is plotted in blue circles and red triangles, respectively. Note that the overall error remains small for a reasonably wide range between roughly $\mu = 0.2$ and 1.8, but increases as μ approaches 0 and 2, indicating that the reservoir begins to fail as the internal dynamics either lose their influence within the reservoir or become too strong.

Last, we examine the case where the reservoir itself is eliminated and disturbances are predicted directly from the state of the system. In this framework, we use a new output matrix W_{out} that is $ND \times ND$ and trained to minimize the error between $\boldsymbol{U} = W_{\text{out}}\boldsymbol{X}$ and H using the same ridge regression procedure as in Eq. (4). Note that this method bypasses the reservoir dynamics and attempts to detect the disturbance solely based on the instantaneous state of the system, forgoing the memory inherent in the reservoir. Using once again the Lotka–Volterra model illustrated in Fig. 2, we consider 200 trials each of disturbance detection with the reservoir approach (reverting to a spectral radius of $\mu = 1.2$) and the bypass approach, i.e., where the reservoir has been eliminated. Similar to the realizations above, at each realization, nodes 3 and 5 were disturbed with the functions $g_3(t) = u_1 \sin(u_2 t) + u_1 \sin(u_3 t)$ and $g_5(t) = u_1 \sin[u_4 \sin(u_2 t)]$, where u_1, u_2, u_3, and u_4 were randomly and uniformly drawn from the intervals $[0.2, 0.4]$, $[1, 2]$, $[\pi/2, \pi]$, and $[\pi, 2\pi]$. In Fig. 11(a), we plot the densities of the base-10 logarithm of the mean squared error for the reservoir method and the bypass method, plotted in blue circles and red triangles, respectively, with unfilled and filled markers corresponding to errors computed for the disturbed and undisturbed nodes, respectively. We find that, whether comparing the error at disturbed or undisturbed nodes, the typical error in the bypass method is roughly 100 times larger than in the reservoir method. In Figs. 11(b) and 11(c), we plot an example of the true (solid blue) and recovered (dashed red) disturbances taken from randomly chosen realizations of the reservoir and bypass methods, respectively.

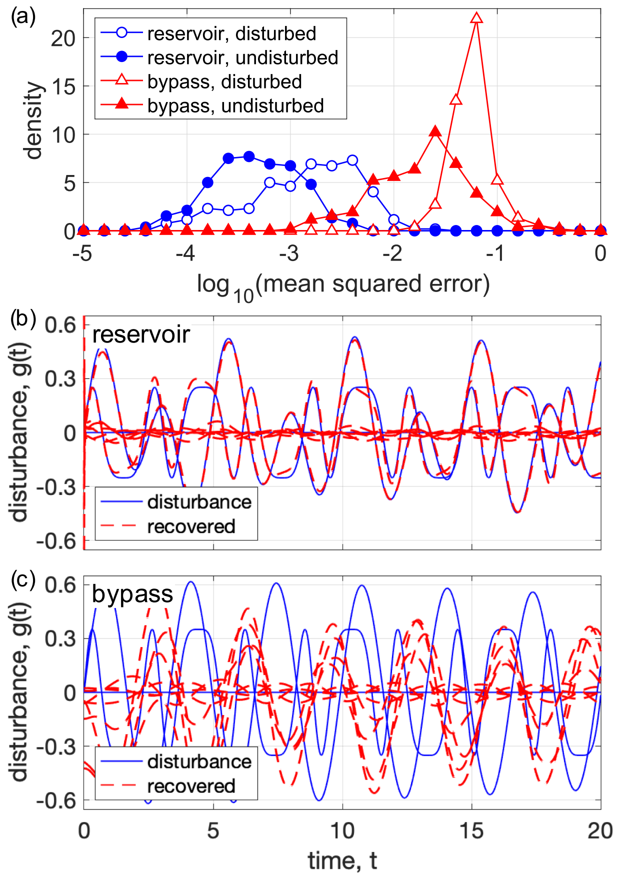

FIG. 11. Reservoir vs bypass: Error analysis. (a) Observed densities of the base-10 logarithm of the mean squared error between true and recovered disturbances for the reservoir (blue circles) and bypass (red triangles) methods, each calculated from 200 realizations. Unfilled and filled markers represent errors for disturbed and undisturbed nodes. Underlying dynamics are given by the Lotka–Volterra model illustrated in Fig. 2 with disturbances made to nodes 3 and 5 using $g_3(t) = u_1 \sin(u_2 t) + u_1 \sin(u_3 t)$ and $g_5(t) = u_1 \sin[u_4 \sin(u_2 t)]$ with u_1, u_2, u_3, and u_4 drawn randomly and uniformly from [0.2, 0.4], [1, 2], $[\pi/2, \pi]$, and $[\pi, 2\pi]$, respectively. (b) and (c) Example of the true (solid blue) and recovered (dashed red) disturbances taken from randomly chosen realizations of the reservoir and bypass methods, respectively.

VI. DISCUSSION

In this work, we have presented a model-free method for identifying disturbances in networks of coupled dynamical systems. This method is based on the machine learning framework of reservoir computing. No knowledge of either the underlying dynamics or the nature of the disturbance itself is assumed. In fact, all that is required is the observed behavior of the system under a known training forcing function.

Using food web and neuronal population models as examples, we demonstrated that this method robustly identifies disturbances to network-coupled dynamical systems using a range of relatively simple forcing functions for training. For certain cases where the dynamics are periodic, we have shown that including a leak in the reservoir dynamics can improve results. Moreover, by implementing a pseudo-parallel reservoir architecture, we illustrated that this method robustly scales with the system size. This parallelization maintains accurate results while providing efficient computation for large networks.

As machine learning techniques become more and more integrated into dynamical systems theory, we emphasize that the technique presented here differs from applications of reservoir computing to nonlinear dynamics in that aim to learn the intrinsic dynamics of a system, and rather align with previous work that aims to infer some extrinsic information, here an external disturbance. We believe that similar techniques may be useful for a wider variety of tasks that go beyond forecasting system behaviors, for example, inferring model parameters, network structures, or the other properties of interactions, e.g., coupling functions.

ACKNOWLEDGMENTS

P.S.S. acknowledges support from the NSF grant (No. MCB-2126177). J.G.R. acknowledges support from the NSF grant (No. DMS-2205967).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Per Sebastian Skardal: Conceptualization (equal); Data curation (equal); Formal analysis (equal); Funding acquisition (equal); Investigation (equal); Methodology (equal); Project administration (equal); Resources (equal); Software (equal); Supervision (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Juan G. Restrepo:** Conceptualization (equal); Data curation (equal); Formal analysis (equal); Funding acquisition (equal); Investigation (equal); Methodology (equal); Project administration (equal); Resources (equal); Software (equal); Supervision (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1]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 Netw. **115**, 100 (2019).
[2]S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, 2019).
[3]Y. Tang, J. Kurths, W. Lin, E. Ott, and L. Kocarev, "Introduction to focus issue: When machine learning meets complex systems: Networks, chaos, and nonlinear dynamics," Chaos **30**(6), 063151 (2020).
[4]K. Nakajima and I. Fischer, *Reservoir Computing* (Springer, New York, 2021).
[5]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," Phys. Rev. Lett. **120**, 024102 (2018).
[6]K. Srinivasan, N. Coble, J. Hamlin, T. Antonsen, E. Ott, and M. Girvan, "Parallel machine learning for forecasting the dynamics of complex networks," Phys. Rev. Lett. **128**, 164101 (2022).
[7]S. Shahi, C. D. Marcotte, C. J. Herndon, F. H. Fenton, Y. Shiferaw, and E. M. Cherry, "Long-time prediction of arrhythmic cardiac action potentials using recurrent neural networks and reservoir computing," Front. Physiol. **12**, 734178 (2021).
[8]S. Shahi, F. H. Fenton, and E. M. Cherry, "Prediction of chaotic time series using recurrent neural networks and reservoir computing techniques: A comparative study," Mach. Learn. Appl. **8**, 100300 (2022).
[9]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**, 121102 (2017).
[10]Q. Zhu, M. Huanfei Ma, and W. Lin, "Detecting unstable periodic orbits based only on time series: When adaptive delayed feedback control meets reservoir computing," Chaos **29**, 9 (2019).
[11]A. Banerjee, J. Pathak, R. Roy, J. G. Restrepo, and E. Ott, "Using machine learning to assess short term causal dependence and infer network links," Chaos **29**, 121104 (2019).
[12]V. Pyragas and K. Pyragas, "Using reservoir computer to predict and prevent extreme events," Phys. Lett. A **384**, 126591 (2020).
[13]L.-W. Kong, Y. Weng, B. Glaz, M. Haile, and Y.-C. Lai, "Reservoir computing as digital twins for nonlinear dynamical systems," Chaos **33**, 033111 (2023).
[14]D. Patel and E. Ott, "Using machine learning to anticipate tipping points and extrapolate to post-tipping dynamics of non-stationary dynamical systems," Chaos **33**, 023143 (2023).
[15]D. Canaday, A. Pomerance, and D. J. Gauthier, "Model-free control of dynamical systems with deep reservoir computing," J. Phys.: Complex. **2**, 035025 (2021).
[16]H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," Science **304**, 78 (2004).
[17]S. Wakabayashi, T. Arie, S. Akita, K. Nakajima, and K. Takei, "A multitasking flexible sensor via reservoir computing," Adv. Mater. **34**, 2201663 (2022).
[18]R. Sakurai, M. Nishida, H. Sakurai, Y. Wakao, N. Akashi, Y. Kuniyoshi, Y. Minami, and K. Nakajima, "Emulating a sensor using soft material dynamics: A reservoir computing approach to pneumatic artificial muscle," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)* (IEEE, 2020), pp. 710–717.
[19]J. G. Restrepo and P. S. Skardal, "Suppressing unknown disturbances to dynamical systems using machine learning," arXiv:2307.03690 (2023).
[20]S. Upadhyaya and S. Mohanty, "Power quality disturbance localization using maximal overlap discrete wavelet transform," in *2015 Annual IEEE India Conference (INDICON)* (IEEE, New York City, 2015), pp. 1–6.
[21]A. T. Mathew and M. N. Aravind, "PMU based disturbance analysis and fault localization of a large grid using wavelets and list processing," in *2016 IEEE Region 10 Conference (TENCON)* (IEEE, New York City, 2016), pp. 879–883.
[22]W.-H. Chen, J. Yang, L. Guo, and S. Li, "Disturbance-observer-based control and related methods—An overview," IEEE Trans. Ind. Electron. **63**, 1083 (2015).
[23]V. H. Ferreira, R. Zanghi, M. Z. Fortes, G. G. Sotelo, R. da Boa Morte Silva, J. C. S. Souza, C. H. C. Guimaraes, and S. Gomes, Jr., "A survey on intelligent system application to fault diagnosis in electric power system transmission lines," Electr. Power Syst. Res. **136**, 135–153 (2016).
[24]D. Wang, X. Wang, Y. Zhang, and L. Jin, "Detection of power grid disturbances and cyber-attacks based on machine learning," J. Inf. Secur. Appl. **46**, 45–52 (2019).
[25]R. Delabays, L. Pagnier, and M. Tyloo, "Locating line and node disturbances in networks of diffusively coupled dynamical agents," New J. Phys. **23**, 043037 (2021).

[26]R. Delabays, L. Pagnier, and M. Tyloo, "Locating fast-varying line disturbances with the frequency mismatch," IFAC-PapersOnLine **55**, 270 (2022).
[27]G. Meurant, *The Ecology of Natural Disturbance and Patch Dynamics* (Academic Press, New York, 2012).
[28]C. Battisti, G. Poeta, and G. Fanelli, *An Introduction to Disturbance Ecology* (Springer, Cham, 2016), pp. 13–29.
[29]T. R. Bewley and S. Liu, "Optimal and robust control and estimation of linear paths to transition," J. Fluid Mech. **365**, 305 (1998).
[30]T. R. Bewley, R. Temam, and M. Ziane, "A general framework for robust control in fluid mechanics," Phys. D **138**, 360 (2000).
[31]J. Verbesselt, R. Hyndman, A. Zeileis, and D. Culvenor, "Phenological change detection while accounting for abrupt and gradual trends in satellite image time series," Remote Sens. Environ. **114**, 2970 (2010).
[32]T. R. Nudell and A. Chakrabortty, "A graph-theoretic algorithm for disturbance localization in large power grids using residue estimation," in *2013 American Control Conference* (IEEE, New York City, 2013), pp. 3467–3472.
[33]H.-W. Lee, J. Zhang, and E. Modiano, "Data-driven localization and estimation of disturbance in the inter-connected power system," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)* (IEEE, New York City, 2018), pp. 1–6.
[34]T. Carroll and L. Pecora, "Network structure effects in reservoir computers," Chaos **29**, 083130 (2019).
[35]A. Shirin, I. S. Klickstein, and F. Sorrentino, "Stability analysis of reservoir computers dynamics via Lyapunov functions," Chaos **29**, 103147 (2019).
[36]E. Del Frate, A. Shirin, and F. Sorrentino, "Reservoir computing with random and optimized time-shifts," Chaos **31**, 121103 (2021).
[37]J. Hart, F. Sorrentino, and T. Carroll, "Time-shift selection for reservoir computing using a rank-revealing QR algorithm," Chaos **33**, 043133 (2023).
[38]C. Nathe, C. Pappu, N. A. Mecholsky, J. Hart, T. Carroll, and F. Sorrentino, "Reservoir computing with noise," Chaos **33**, 041101 (2023).
[39]J. C. Canfield, *Active Disturbance Cancellation in Nonlinear Dynamical Systems Using Neural Networks* (University of New Hampshire, 2003).
[40]J. Hofbauer, *Evolutionary Games and Population Dynamics* (Cambridge University Press, 2012).
[41]H. R. Wilson, "Hyperchaos in Wilson-Cowan oscillator circuits," J. Neurophysiol. **122**, 2449 (2019).