Model-free inference of unseen attractors: Reconstructing phase space features from a single noisy trajectory using reservoir computing

Cite as: Chaos **31**, 103127 (2021); https://doi.org/10.1063/5.0065813
Submitted: 05 August 2021 • Accepted: 25 September 2021 • Published Online: 26 October 2021

André Röhm, Daniel J. Gauthier and Ingo Fischer

Model-free inference of unseen attractors: Reconstructing phase space features from a single noisy trajectory using reservoir computing

Cite as: Chaos **31**, 103127 (2021); doi: 10.1063/5.0065813
Submitted: 5 August 2021 · Accepted: 25 September 2021 ·
Published Online: 26 October 2021

View Online Export Citation CrossMark

André Röhm,[1,2,a)] Daniel J. Gauthier,[3,4] and Ingo Fischer[1,b)]

AFFILIATIONS

[1]Instituto de Física Interdisciplinar y Sistemas Complejos, IFISC (CSIC-UIB), Campus Universitat Illes Balears, E-07122 Palma de Mallorca, Spain
[2]Department of Information Physics and Computing, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
[3]Department of Physics, The Ohio State University, 191 West Woodruff Ave., Columbus, Ohio 43210, USA
[4]ResCon Technologies, LLC, PO Box 21229, Columbus, Ohio 43221, USA

[a)]**Author to whom correspondence should be addressed:** roehm@g.ecc.u-tokyo.ac.jp
[b)]ingo@ifisc.uib-csic.es

ABSTRACT

Reservoir computers are powerful tools for chaotic time series prediction. They can be trained to approximate phase space flows and can thus both predict future values to a high accuracy and reconstruct the general properties of a chaotic attractor without requiring a model. In this work, we show that the ability to learn the dynamics of a complex system can be extended to systems with multiple co-existing attractors, here a four-dimensional extension of the well-known Lorenz chaotic system. We demonstrate that a reservoir computer can infer entirely unexplored parts of the phase space; a properly trained reservoir computer can predict the existence of attractors that were never approached during training and, therefore, are labeled as *unseen*. We provide examples where attractor inference is achieved after training solely on a single noisy trajectory.

Reservoir computing is a brain-inspired machine learning scheme that can be used to mimic dynamical systems. Reservoir computers can be trained to learn the characteristics of a target dynamical system purely from a sample time series. In particular, properly trained autonomous reservoir computers can act as surrogate systems while still preserving many properties of the original ground truth such as the largest Lyapunov exponents, embedded unstable periodic orbits, or correlation measures. Importantly, dynamical systems can exhibit more than just a single stable long-term behavior, called an attractor. A common scenario is the existence of pairs of symmetric solutions, but more complex co-existences can also often be found. Systems with multiple attractors are called multistable. Here, we provide examples where a reservoir computer is able to learn the various attractors of a multistable system. We feed the reservoir just a single noisy trajectory of one of the attractors, while the other attractors remain outside of the training data range. Then, in separate autonomous operation, the trained reservoir is able to reproduce and, therefore, infer the existence and shape of these *unseen* attractors.

I. INTRODUCTION

Reservoir computing is a brain-inspired machine learning technique that was popularized by the works of Maass *et al.*[1] and Jaeger[2] in the form of liquid state machines and echo state networks, respectively. At its core, a reservoir computer usually consists of three elements: a fixed input layer; a fixed dynamical system with a high-dimensional phase space, called the "reservoir;" and a linear,

trainable readout layer. In particular, the reservoir can be a recurrent artificial neural network (RNN) with fixed connection weights, in which case, the system is still referred to as an echo state network.

Regardless of the type, the reservoir introduces transformations and memory for the input sequence by virtue of responding to an input while being in a state induced by the previous input. By reading out the many degrees of freedom of the reservoir and potentially performing low-order polynomial transformations, the reservoir provides a large number of transformations of the input sequence, i.e., a multitude of responses. As opposed to other RNN schemes, the readout layer is the only part that is trained in reservoir computing. A linear readout layer is often sufficient, given a nonlinear reservoir, and can easily be optimized via a straightforward regression on a set of training data. It has recently been shown mathematically that linear reservoirs with nonlinear readouts can also provide universal computational properties.[3,4]

It was soon discovered that reservoir computers are highly capable at predicting chaotic time series, improving accuracy by orders of magnitude compared to previous methods.[5] By feeding the reservoir a chaotic time series and choosing the target to be the next point of the chaotic trajectory, the reservoir computer learns to perform the so-called one-step-ahead prediction. Intriguingly, after training for one-step-ahead prediction, such a reservoir can be put into a "closed loop configuration," where its *own* prediction can be used as the next input. In this way, the reservoir computer becomes an autonomous dynamical system that can accurately continue the original data. Generally, in a chaotic attractor, the predicted time series does not agree with the specific original training data forever, as even tiny differences will eventually lead to divergence of trajectories. However, the autonomously created time series can reproduce the general structure of the chaotic attractor it was trained on even for long time scales. This is sometimes referred to as the "climate prediction" property, as opposed to the short-term prediction error, which analogously is referred to as "weather prediction" property. It is worth emphasizing that the reservoir is reproducing the source system in a model-free manner, i.e., without any knowledge about the origin of the training data.

Chaotic time series prediction using reservoir computing continues to be an active field of research with several noteworthy results. Pathak *et al.*[6] showed that properly trained reservoir computers can be used to reconstruct the largest Lyapunov eigenvalues inside the chaotic attractor. Chen *et al.*[7] showed that such reservoir computers also reproduce several typical geometric metrics. Estebanez *et al.*[8] demonstrated how noise can be used to improve the long-term attractor reconstruction, and Zhu *et al.*[9] showed how delayed-feedback control can be combined with reservoir computing to find the unstable periodic orbits embedded inside the chaotic attractor. Furthermore, unstable fixed points can be found using reservoir computing, even when the trajectory never visits them during training.[4,10] Kim *et al.*[11] recently showed that a reservoir computer can correctly predict bifurcations if training is done with the bifurcation parameter as an input explicitly fed to the reservoir. Remarkably, they only used training trajectories taken from time series data generated below the bifurcation point to teach the reservoir the influence of the bifurcation parameter. These results hint at the ability of a properly trained reservoir computer to reproduce the phase space structure of the original system accurately and even its control parameter dependence when operated in an autonomous mode.

We demonstrate that the ability of phase space reconstruction can be exploited even further. We show that a properly trained reservoir computer can not only reconstruct the attractor it is trained on but also can even infer other unseen co-existing attractors in the system's phase space and reconstruct their structure. By unseen co-existing attractors, we specifically mean those attractors in a multistable system whose basin of attraction was never reached during training so that no direct traces of them can be found in the original time series data. To this end, we test the ability of a reservoir computer to infer unseen attractors for a four-dimensional chaotic extension of the Lorenz system with co-existing attractors in two scenarios. The first scenario comprises a torus solution co-existing with a pair of symmetric limit cycles. In the second scenario, a pair of symmetric chaotic attractors coexists with a torus.

Attractor reconstruction is arguably related to the field of nonlinear system identification, which aims to derive the governing equations from a sample set of time series data. In particular, techniques such as Sparse Identification of Nonlinear Dynamics (SINDy),[12] ResNet-based approaches,[13] auto-regressive moving average (ARMA) models, and variants (ARMAx and NARMAx) are capable of producing accurate model descriptions in some cases.[14] However, all of these either require previous knowledge about the general structure of the system or multiple sets of training trajectories. Moreover, those that only estimate the vector field would have to be combined with an integrator to facilitate one-step-ahead prediction.

In contrast, we show that reservoir computers, operated in the autonomous mode, can serve as model-free surrogates of target systems even outside of their training region with minimal input, i.e., they can learn to reconstruct unseen attractors after learning from a single, noisy trajectory. Thus, reservoir computers can satisfactorily mimic target systems in cases where the training data are noisy, not a lot of training data are available, and nothing concrete about the shape of the underlying differential equation is known. Here, the reservoir computers do not learn the governing equations of the original system, instead they learn how to integrate and propagate along trajectories. Thus, the reservoir computers are learning the phase space flow without formulating any intermediate model. From recent studies, it is known that attractor reconstruction can sometimes fail even for parameter-sets optimized for one-step-ahead prediction.[15,16] This also applies to the cases presented here, where a reservoir computer infers the existence of other attractors. Lacking a way to verify the prediction on the original system, this failure mode is harder to detect because the original data does not contain any sampling from the unseen attractors. Obtaining a quantitative estimate of the quality of the attractor reconstruction without knowing the target system remains an open problem.

In the following, we first introduce the model used and the target system chosen. We then analyze the reservoir computers' ability to reproduce unseen attractors. We find that for a set of parameters the reservoir computer succeeds in predicting the existence of a torus and symmetric limit cycle with low errors. In the second scenario, involving two chaotic attractors and a torus, it is more difficult to succeed. We analyze cases of partial success and discuss what the current limitations are.

II. THEORY

A. Echo state network

We use a continuous time version of an echo state network based on ordinary differential equations similar to those used by Lu *et al.*[17] The reservoir computer consists of a network of N coupled real-valued nodes with the total state $X \in \mathbb{R}^N$ describing all nodes evolving according to

$$\dot{X} = -X + \tanh\left(W_{\text{res}}X + GW_{\text{in}}u(t) + B\right), \tag{1}$$

where the random matrices $W_{\text{res}} \in \mathbb{R}^{N\times N}$ and $W_{\text{in}} \in \mathbb{R}^{N\times U}$ describe the internal and the input weights, respectively. The input gain is given by scalar G. The time-varying input $u(t) \in \mathbb{R}^U$ is a step-function with period θ, with the amplitude of each plateau corresponding to one point of data of the discrete source series $u_k \in \mathbb{R}^{U\times K}$. $B \in \mathbb{R}^N$ is a random bias vector. The network and input matrices are sparse, and the details of the random initialization and the list of all parameters are given in Appendix A.

The state X of the network is periodically recorded at intervals θ and used to construct the state matrix $S \in \mathbb{R}^{K\times N+1}$. We add a bias column into the state matrix such that a row S_k is finally given by

$$S_k = [X_1(t_k), X_2(t_k), \ldots, X_N(t_k), 1], \tag{2}$$

where $t_k = T_w + k\theta$ with T_w being the "washout" or "warmup" time used to remove the influence of the starting state of the reservoir. The prediction $Y \in \mathbb{R}^{K\times U}$ of the reservoir is then given by

$$Y = SW_{\text{out}}. \tag{3}$$

A training series of K_{training} elements is used to drive the system to generate S_{training}, which, in turn, is used to determine the output weights $W_{\text{out}} \in \mathbb{R}^{N+1\times U}$. The error between the prediction Y and the known true targets $\hat{Y}$ is minimized using an L_2-norm. To reduce the amount of over-fitting, we use Tikhonov regularization as described in Appendix C. The regularization strength η is an important parameter as it controls the level of detail the training data are approximated to. High η will fit the training data too crudely, rendering attractor inference impossible. Conversely, low η will lead to over-fitting and poor generalization.

Once the reservoir computer is successfully trained, we use it to probe other parts of the phase space of the target system. To this end, we feed the reservoir computer the beginning of a different ground-truth transient, which for now is required. We then observe how it evolves in the autonomous mode. See Appendix C for details.

B. Target system by Li and Sprott

We study the properties of unseen attractors and the ability of a reservoir computer to infer their existence. As our target dynamical system, we use a four-dimensional extension of the Lorenz attractor as proposed by Li and Sprott,[18] based on earlier work by Gao and Zhang,[19] whose dynamics is given by

$$\dot{x} = y - x + \sigma\xi_x, \tag{4}$$

$$\dot{y} = -xz + u + \sigma\xi_y, \tag{5}$$

$$\dot{z} = xy - a + \sigma\xi_z, \tag{6}$$

$$\dot{u} = -by + \sigma\xi_u, \tag{7}$$

which has only two parameters a and b. Here, we add additive independent identically distributed Gaussian noise terms ξ_i with mean 0 and unit variance, $E[\xi_i(t)\xi_j(t')] = \delta_{ij}\delta_{tt'}$. For the sake of simplicity, different variables share the same noise strength σ. This simplification is acceptable because all variables share the same order of magnitude. For $\sigma = 0$, the system is noise-free and deterministic. We chose this system because it shows coexisting attractors of different complexities depending on parameters. This allows us to test the ability of reservoir computers to infer the existence of unseen attractors in various scenarios involving periodic, quasiperiodic, and chaotic attractors.

By initializing the system (4)–(7) in different initial conditions, we can reach different attractors, as each trajectory will eventually reach one of the stable attractors. The volume of phase space from which trajectories lead to a certain attractor is called its basin of attraction. Notably, the system (4)–(7) never features any fixed points, and hence, the attractors are called hidden and the basins of all attractors studied in this work are fractal.[18]

C. Error estimates

We use a quantitative measure for the quality of attractor reconstruction. For the reconstructed and target attractors, we take the time average for each variable $\langle x\rangle$, $\langle y\rangle$, $\langle z\rangle$, $\langle u\rangle$, and the time average of the absolute values $\langle|x|\rangle$, $\langle|y|\rangle$, $\langle|z|\rangle$, $\langle|u|\rangle$. The absolute values help differentiate when averages become 0, such as for periodic states centered around the origin.

We determine the differences Δ between prediction x and the ground-truth $\tilde{x}$, normalizing by the average ground-truth absolute averages

$$\Delta_x = \frac{\langle x\rangle - \langle\tilde{x}\rangle}{\langle|\tilde{x}|\rangle}, \tag{8}$$

$$\Delta_{|x|} = \frac{\langle|x|\rangle - \langle|\tilde{x}|\rangle}{\langle|\tilde{x}|\rangle} \tag{9}$$

and similarly for y, z, and u. From this, we calculate an error estimate Δ_{att} for the target attractor as the root of the sum of all squares

$$\Delta_{\text{att}} = \sqrt{\sum_{i=\{x,y,z,u\}} \Delta_i^2 + \Delta_{|i|}^2}. \tag{10}$$

While the quantitative error Δ_{att} does not capture the full picture, it allows us to easily discriminate between (partially) successful and failed attractor inference. To judge the power of a particular reservoir computer for a given scenario, we sum Δ_{att} for all existing target attractors to the total error Δ_{tot} via

$$\Delta_{\text{tot}} = \sqrt{\sum \Delta_{\text{att}}^2}. \tag{11}$$

The applicability of other geometric error measures can also be considered, in particular, the symmetric Hausdorff distance and the average Euclidean distances per point between the real and inferred

attractor. However, the former is unsuited because it only measures the worst point. The latter suffers from requiring too many data points. Our sets sample the chaotic attractors too thinly, poorly covering rare trajectories, whose points, in turn, dominated the average Euclidean distance. In contrast, the simple error measure Δ_{att} of Eq. (10) uses averages and is quite robust, even for small dataset sizes.

We optimize the meta-parameters of the reservoir computers with the help of this error measure Δ_{att}. In this context, a meta-parameter is any parameter of the reservoir, i.e., a parameter that does not refer to the Li–Sprott oscillatory system. Some meta-parameters differ between simulations for the chaotic vs the limit cycle scenarios. We obtain optimal values via a rough grid search for the number of nodes $N = 300$, the network sparsity $\rho = 0.1$, input strength $G = 0.3$ or $G = 0.01$, the regularization strength $\eta = 10^{-3}$ or $\eta = 10^{-5}$, the bias strength $b = 1.0$ or $b = 3.0$, the time per input $\theta = 2.5$, the washout time $T_w = 2500$ and the maximum eigenvalue of the reservoir $\text{Re}(\lambda)_{\text{max}} = 0.95$ or $\text{Re}(\lambda)_{\text{max}} = 0.99$.

III. RESULTS

We first focus on the parameters $a = 2.0$ and $b = 0.8$ in the noise-free case $\sigma = 0$, for which the target system possesses three stable solutions: two periodic limit cycles and a quasiperiodic torus as shown in Fig. 1. The two limit cycles are symmetric with respect to the transformation $(-x, -y, z, -u)$. For these parameters, the limit cycles have oscillations on a time scale of 15 units, while the slow oscillation in the torus lasts 30 and the fast one 5 units, respectively.

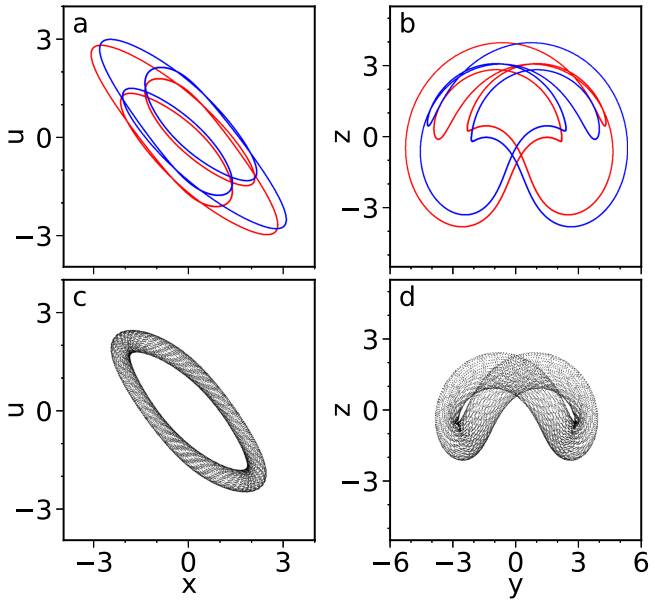

FIG. 1. Solutions of the Li–Sprott equation system for $a = 2.0$, $b = 0.8$, and $\sigma = 0$. Panels (a) and (c) show x–u projection, panels (b) and (d) a z-y projection. A pair of symmetric limit cycles [red and blue lines in (a) and (b)] coexists with a torus [black dots in (c) and (d)].

We sample the training time series with a step size of 0.3 or about 15 points per fast oscillation of the torus.

The starting values for the transients used for training and details of numerical integration of the source system are described in Appendix B. We always train on trajectories with 11 000 data points that stay within one basin of attraction. Of these, 1000 points are used as initial warmup time T_w.

We then train a continuous time echo state network of Eq. (1) with $N = 300$ on the Li–Sprott system with the phase space and attractors as shown in Fig. 1. We designate one of the two limit cycles as our training attractor and initialize a transient that converges to this limit cycle. Using this transient, we drive the reservoir computer, recording the activation of all the nodes. As per standard approach for attractor reconstruction, our training target is a one-step-ahead prediction. The optimal output matrix W_{out} is found and used subsequently.

For our goal of having a system that can find unseen attractors, we use the autonomous operation of the reservoir computer. For this, we feed the reservoir its own prediction. In our case, additional care must be taken to avoid a training failure. First, it is typical that the beginning of the training data does not directly translate into the state matrix S_{training}. In fact, we want the state X of our reservoir to be only dependent on the drive signal $u(t)$. Thus, to avoid any influence of the initial state $X(0)$ of the reservoir on training, the first T_w of $u(t)$ is used to "washout" or "warmup" the reservoir. We use 1000 of the 11 000 training data points for this. But the training limit cycle is highly attracting. Therefore, any reasonably long "washout" period T_w leads to the loss of all information contained in the initial transient of the Li–Sprott system. The reservoir will effectively be trained only on the target limit cycle itself.

Second, when the training data is too low dimensional, the reservoir does not get to "see" the full shape of the dynamical flow. Even for relatively short T_w, the remaining training data only consists of the one-dimensional limit cycle. Even when training succeeds in achieving a low fitting error on the training set, the system is susceptible to being unstable in the autonomous mode. Because certain phase space directions might never be seen during training, the output of the reservoir computer in the autonomous mode tends to diverge in those directions.

Fortunately, these problems can be overcome with a modification that makes training both more stable and makes the whole procedure more appropriate for realistic problems, by the inclusion of noise in the training data. When $\sigma > 0$, the training data produced by the Li–Sprott system (4)–(7) is in a sense permanently transient. Even small noise is enough to allow the trajectory to at least partially diverge from the pure limit cycle, effectively sampling all directions in the four-dimensional phase space. While not explored in detail, there is likely a trade-off between training length, noise strength, and regularization that influences the performance of the reservoir. We ensure that the noise does not lead to any attractor hopping in the training dataset by visual inspection. We also tried using additive noise combined with a noise-free time series as has been studied previously,[8] but did not achieve success, likely due the fact that we are not only targeting chaotic attractors.

In comparison to Kim *et al.*,[11] we do not use multiple parallel training trajectories. Instead, we only train on a single, noisy trajectory. Also, compared to their work, we use a much coarser sampling

of the original time series as they used every point of the source data without any sampling and even included the Runge–Kutta auxiliary terms.

We always feed the reservoir all four variables of the Li–Sprott system both during training and the start of the autonomous mode, i.e., in this work, there are no unseen degrees of freedom. Therefore, the reservoir obtains the full information about the target system in each time step. In principle, this should imply that no memory is needed and that removing the recurrent connections by setting $W_{res} = 0$ in an approach similar to an extreme learning machine could suffice.[20] We performed a few preliminary investigations with $W_{res} = 0$ and find that performance worsened. However, a detailed study of such an approach is outside the scope of this work. In general, we believe that the results presented in this paper can be extended to partially observed target systems in which case memory would be necessary. Furthermore, our chosen architecture is compatible with experimental reservoir computing, where the memory of the reservoir is an important aspect.

A. Inferring the torus and symmetric limit cycle

Figure 2 shows a case of successful training and attractor inference: The system is trained on a single noisy trajectory of with $K_{training} = 10^4$ points and noise strength $\sigma \simeq 6 \times 10^{-3}$. For testing, we feed the system the first 1000 points of a ground-truth reference trajectory (see Appendix C for details) for each of the three attractors (top, middle, and bottom rows in Fig. 2) and then put it into the autonomous mode generating 10 000 points on its own. We always use noise-free ($\sigma = 0$) transients for initializing the reservoir in the autonomous mode and as targets.

The red dots in Fig. 2 represent the reconstructed attractor by the reservoir, while the black dots show the true attractors of the Li–Sprott system, the latter are almost completely covered by the former. The system learns the attractor it is trained on [Figs. 2(a) and 2(b)] with a low error of $\Delta_{att} = 8.4 \times 10^{-3}$. Moreover, it can also predict with high fidelity the existence and full shape of the second, symmetric limit cycle ($\Delta_{att} = 7.6 \times 10^{-3}$) and of the torus [Figs. 2(e) and 2(f), $\Delta_{att} = 5.6 \times 10^{-2}$]. The total error in this case was $\Delta_{tot} \approx 0.06$.

All attractors have complex shapes. It had been shown that a reservoir computer trained on a chaotic system can implicitly learn the position of surrounding and defining *unstable* fixed points and the implicit unstable periodic orbits within.[4,9,10] Figure 2 clearly shows a case where we can go beyond this by demonstrating how an autonomous reservoir computer can infer entirely separate attractors in a target system. In particular, because the attractors of an autonomous system are always disjoint, this means that the system has to infer the properties of parts of the phase space that were neither part of the training data nor are connected to it via any trajectories, unlike unstable structures which might be connected via heteroclines. Here, the reservoir computer must make a plausible inference of the global phase space flow of the dynamical system from local knowledge.

It is important to note that there is quite a variation in the quality of the attractor reconstruction depending on both the target as well as the randomly generated reservoir. In the following, we address a more difficult parameter region of the Li–Sprott system and demonstrate how partial successes and inference failure manifest themselves in such cases.

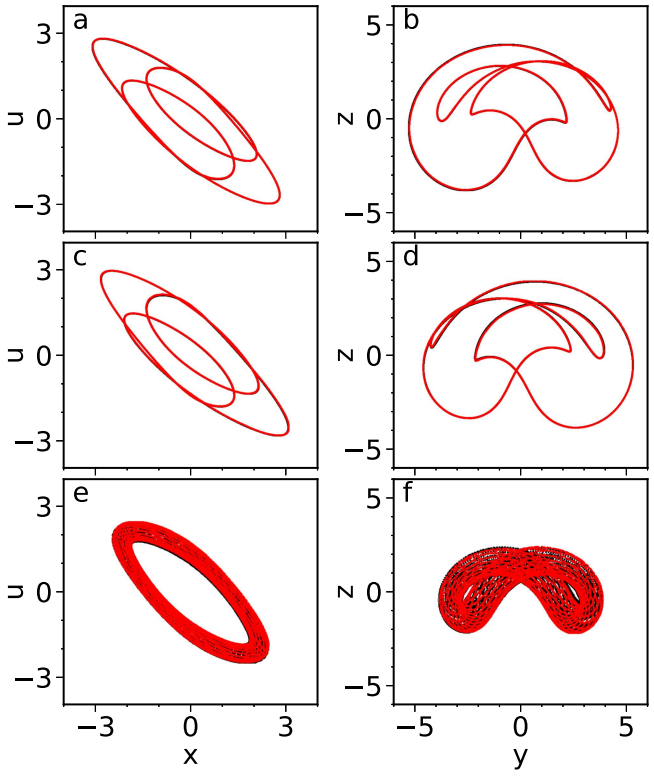

FIG. 2. An example of successful attractor inference for $a = 2.0$ and $b = 0.8$ of the Li–Sprott system [Eqs. (4)–(7)]. The two columns show two projections of the four-dimensional system. The original system has three attractors (black dots): a pair of symmetric limit cycles (a)–(d) and a torus (e) and (f). A reservoir computer in the autonomous mode is used to reproduce the shape ("climate") of these attractors (red points). Training is done only on the first limit cycle (a) and (b). The existence and shape of the other two attractors (c)–(f) is inferred by the reservoir. $G = 0.3$, $b = 1.0$, $\eta = 1e-3$, $\mathrm{Re}(\lambda)_{max} = 0.95$, $\theta = 2.5$. (a) and (b) $\Delta_{att} = 8.4 \times 10^{-3}$, (c) and (d) $\Delta_{att} = 7.6 \times 10^{-3}$, and (e) and (f) $\Delta_{att} = 5.6 \times 10^{-2}$.

B. Training in chaos and quasiperiodic dynamics

Here, we change the target of training. Using the Li–Sprott system with $a = 6.0$ and $b = 0.1$, we enter a regime where two symmetric chaotic attractors coexist with a torus. Once again, we train the system on one of the attractors using a single noisy trajectory. We find that attractor reconstruction and inference in this regime are significantly harder. In particular, we never achieve a total error $\Delta_{tot} < 2$, which is almost two orders of magnitude larger than in the previous case.

When training on one of the chaotic attractors, the system has to infer the existence of its symmetric counterpart and a torus. Despite extensive numerical investigation, we do not find a case where the reservoir computer fully succeeds. Figure 3 shows one

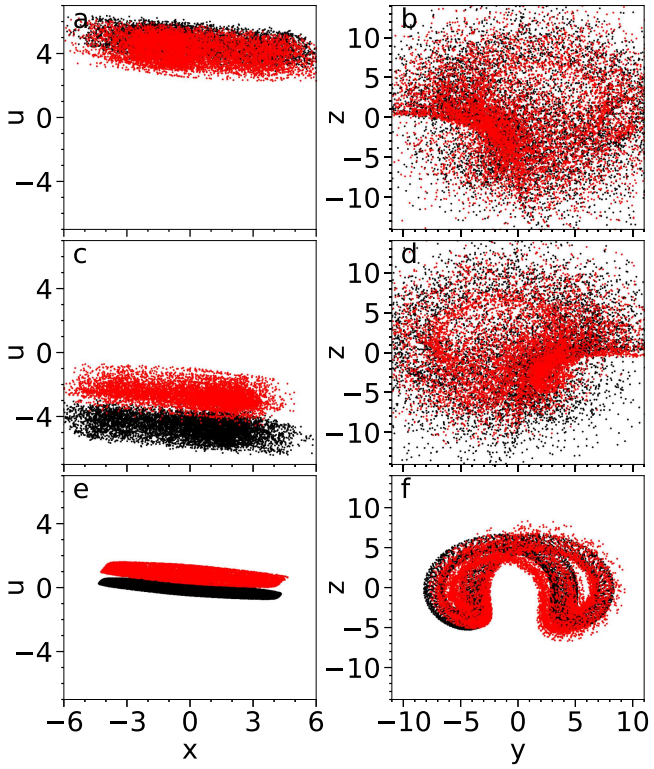

FIG. 3. Partially successful inference for $a = 6.0$ and $b = 0.1$. The two columns show two projections of the four-dimensional system. The original system has three attractors (black dots): a pair of symmetric chaotic regions (a)–(d) and a torus (e) and (f). A reservoir computer in the autonomous mode is used to reproduce these attractors (red points). Training is done only on the positive-u chaotic region (a) and (b). The existence and shape of the other chaotic attractor (c) and (d) and torus (e) and (f) is inferred by the reservoir. $G = 0.01$, $b = 3.0$, $\eta = 1e-5$, $\mathrm{Re}(\lambda)_{\max} = 0.99$, $\theta = 2.5$. (a) and (b) $\Delta_{\mathrm{att}} = 0.14$, (c) and (d) $\Delta_{\mathrm{att}} = 0.65$, (e) and (f) $\Delta_{\mathrm{att}} = 2.3$.

of the most successful results with an error of $\Delta_{\mathrm{tot}} = 2.4$. The system is able to learn the chaotic region it is trained on [Figs. 3(a) and 3(b), $\Delta_{\mathrm{att}} = 0.14$] and it correctly infers the existence and type of the two other attractors, as depicted in Figs. 3(c)–3(f). These other reconstructed attractors show obvious deviations. For the reconstructed symmetric chaotic region, the dynamics of the x, y, and z variables are, in fact, almost accurately predicted with errors Δ_i at most 0.18. However, the u variable shows a persistent positive offset from the ground truth, as can be seen in Figs. 3(c) and 3(e), and this is reflected by higher errors $\Delta_u \approx -\Delta_{|u|} \approx 0.43$. Similarly, the torus [Figs. 3(e) and 3(f)] is mismatched in the u-dimension with $\Delta_u \approx 0.7$. In addition, the shape is distorted increasing the error of the average absolute values $\Delta_{|x|} \approx 0.6$, $\Delta_{|y|} \approx 1.0$, $\Delta_{|z|} \approx 1.9$, and $\Delta_{|u|} \approx -0.3$.

The deviation in the u variable for the unseen attractors is likely due to the large separation between the training and target attractor; the two chaotic regions are clearly separated from the torus and from each other in the u-variable. We assume that this large separation makes an accurate inference harder as compared to the case of the intertwined limit cycles and torus studied earlier. The reservoir is required to correctly infer the phase space flow over longer distances, which accumulates the visible deviations between ground truth and prediction in Figs. 3(e) and 3(f).

Furthermore, the different time scales cannot be ignored in this case. While the limit cycle and torus dynamics of Sec. III A allowed for a sampling that was reasonably adequate for all oscillations involved, this is not the case here. In particular, the chaotic dynamics show both slow oscillations on a time scale of 100 and fast oscillations on the order of 3. Similarly, the torus contains slow oscillations around 6 and fast oscillations of order 120. This is a separation of almost two orders of magnitude, which makes choosing a suitable sampling rate a difficult compromise. We chose a sampling interval of 0.2, mostly adapted to the fast oscillations. Nevertheless, even in this more difficult case, the system correctly predicts the type, shape, and rough position of the unseen attractors. If knowing the exact u position and torus shape is not required, the reservoir successfully predicts the unseen attractors.

We find many cases of partial success and failure. Two types of obvious failures, in particular, are detectable even in a model-free setting. First, if the reservoir fails to reproduce the training dataset, it is safe to assume that its predictions cannot be considered dependable. We see this type of failure remarkably often, and the fact that some randomly generated reservoir topologies work worse than others is a known issue.[15] A second detectable failure shows the attractor inference time series veering off toward infinity or settle at unreasonably large values. Reservoirs with such unphysical predictions are also easily discarded.

To quantify these errors rates, we investigate the variability of the reservoir performance. Keeping all parameters and training targets fixed at the values used for Fig. 3, we simulate a set of 2000 random reservoir topologies. Out of these 2000 simulations, over 1300 show a deviation of $\Delta_i \geq 100$ in at least one variable. This indicates that the trajectory failed to converge. Out of the remaining roughly 700 reservoirs, only 240 stay below an error of $|\Delta_i| < 2$ in every metric. These reservoirs show different degrees of success.

We show the subset of simulations with the lowest total error Δ_{tot} in Fig. 4. The histogram shows a clear bimodal distribution. While the different predictions do not cleanly fall into classes, after investigating the source of this bimodality, we conjecture that the first peak is related to predictions that approximate the targets as good as possible. The second peak corresponds to cases in which inference fails and only the training region is predicted to be stable, i.e., the multistability cannot be inferred.

Ultimately, the error measures Δ always compress information. When there is a significant deviation between ground truth and prediction, many cases of partial success arise with different qualities. We show an additional example to highlight how partial success can manifest itself.

Figure 5 shows an example where inference deviates from the ground truth by one attractor exhibiting a different type and another not being detected. Figures 3 and 5 only differ in their randomly generated topology. As in the successful trial, the reservoir succeeds in reproducing the training attractor [Figs. 5(a) and 5(b)] with a low error of $\Delta_{\mathrm{att}} = 0.09$ and, therefore, barring additional information, the attractor inference of both Figs. 3 and 5 appears equally valid.

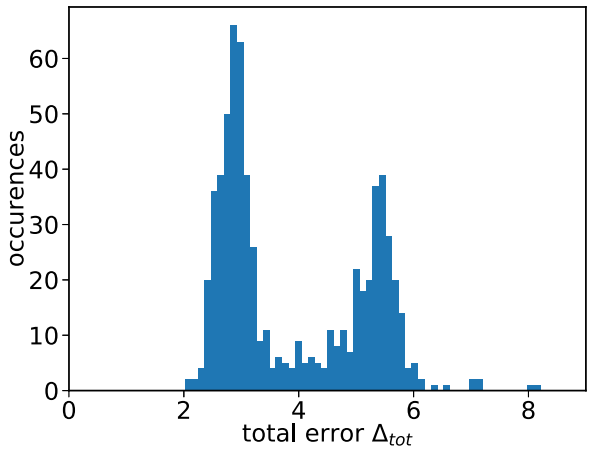

FIG. 4. Histogram of the distribution of the total error Δ_{tot} for a set of 2000 randomly generated reservoir topologies, while the other parameters are fixed. Note that more than 1300 simulations exhibit large errors Δ_{tot} and lie outside of the range depicted here.

Nevertheless, the prediction of the two unseen attractors is less successful for Fig. 5. The total error is slightly larger with $\Delta_{tot} \approx 2.5$. Inference of the torus is not fully achieved and instead a limit cycle is predicted [Figs. 5(e) and 5(f), $\Delta_{att} = 2.1$]. We still call this a partial success. When reconstructing the dynamic behavior of a completely unknown dynamical system, the information that there might be an object of interest in a particular phase space region is still valuable information. The reservoir can at least predict the existence of some attracting structure despite failing to predict the torus. Similarly, we can sometimes observe cases where a torus or limit cycle is predicted with slightly distorted shape. Furthermore, this limit cycle roughly follows the outline of the torus, indicating that it correctly reflects some truth about the underlying phase space. It might also be possible that this limit cycle exists in the original system but is only slightly unstable.

In contrast, the inference of the chaotic region fails for Figs. 5(c) and 5(d) with $\Delta_{att} = 1.5$. The reservoir does not even detect the existence of an attracting structure and converges to the same attractor as Figs. 5(e) and 5(f). In this case, there is no obvious trace in the reconstructed reservoir time series about this symmetrical chaotic region.

In general, we observe that the behavior in cases of such partial successes and failures exhibits a broad range of types. It ranges from underestimating the width of a torus over simplifications of attractor shapes to predicting the entirely wrong class of attractor, e.g., a limit cycle in place of the torus. Furthermore, the quality of reproduction of one of the unseen attractors does not directly translate to the quality of the other.

For now, we have no method for preventing this kind of failure. If a method could be devised that can judge the quality of inferred attractors, one could use this to discriminate between successful and failing reservoirs. One potential avenue is a scenario for which one does know two attractors of the target system: In this case, one can use one of them as a training dataset and the other as an independent

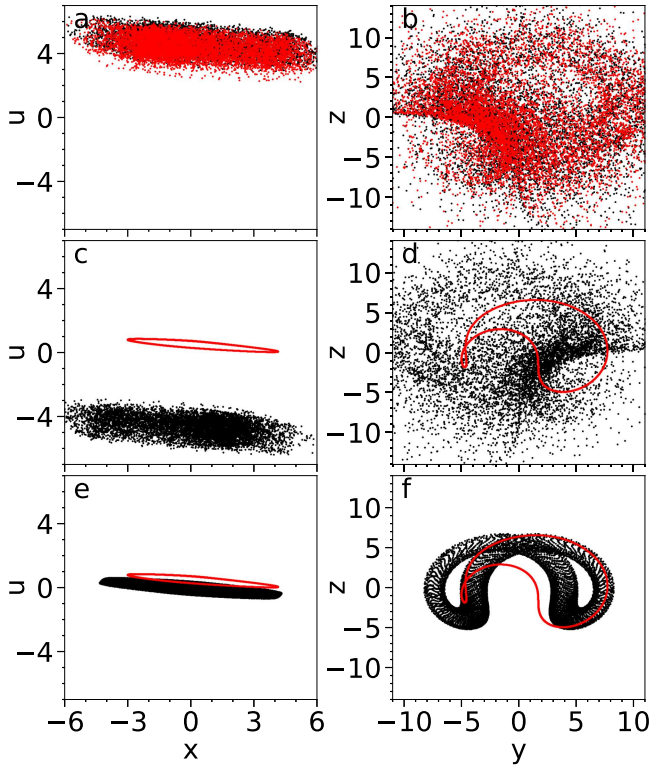

FIG. 5. Partial inference success and failure for $a = 6.0$ and $b = 0.1$. The two columns show two projections of the four-dimensional system. The original system has three attractors (black dots): a pair of symmetric chaotic regions (a)–(d) and a torus (e) and (f). A reservoir computer in the autonomous mode is used to reproduce these attractors (red points). Training is done only on the positive-u chaotic region (a) and (b). The existence and shape of the other chaotic attractor (c) and (d) is not correctly inferred by the reservoir, despite success on the training data and instead of the torus (e) and (f) a limit cycle is predicted. $G = 0.01$, $b = 0.1$, $\eta = 1 \times 10^{-5}$, $\mathrm{Re}(\lambda)_{max} = 0.99$, $\theta = 7.5$. (a) and (b) $\Delta_{att} = 4.5 \times 10^{-2}$, (c) and (d) $\Delta_{att} = 1.4$, (e) and (f) $\Delta_{att} = 2.1$.

test. For systems with only a single training dataset, independent error estimation remains an open question for now.

A final open question regards the role of noise in the training data: minute details of the phase space flow that might be important for long range inference are washed out by the noise. We, therefore, tried to reduce the noise as much as possible. However, a certain level of noise is necessary for our scheme to succeed. Systematically investigating the relationship between necessary noise levels (as compared to computational noise levels), training series length, inference distance, and regularization strength is an interesting challenge for future work to give insights into these important aspects.

IV. DISCUSSION

We report the successful use of an autonomously operated reservoir computer for attractor reconstruction. This includes the reconstruction of not only the training attractor but also the

attractors that the system never sees during training. Moreover, we perform training using only a single noisy trajectory. This demonstrates that reservoir computers are powerful tools for model-free system analysis and attractor prediction, as there are no assumptions put into the reservoir about the target system.

We show how the system is able to correctly predict the existence and shape of a pair of symmetric intertwined complex limit cycles and a torus in a four-dimensional extension of the Lorenz system. We then explore more difficult scenarios involving two chaotic regions and a torus. For this problem, we obtain partial success. Even for cases when we cannot find reservoirs that fully succeeded in attractor inference, some useful partial information is still extracted such as the approximate location of an attractor or just the existence of "some kind" of attractor.

Several open questions remain to further develop this approach. First, and foremost, the fraction of randomly generated reservoirs that fail is quite high. For non-optimized parameters, it can be more than 95%. Even for optimized networks, the percentage remains high. For the parameters used in Fig. 2, about 1500 out of 2000 randomly generated reservoirs cannot predict the existence of the symmetric limit cycle solution. The reservoir might simply fail to predict any kind of other attractor in the system. Some failures are detectable even if nothing else is known about the target system. Some failures, however, produce plausibly looking but false reconstructions. Finding methods for quality control of the reproduced attractors remains an open problem.

The inference errors are higher in the second investigated case involving two chaotic regions and a torus. Here, we observe much larger deviations Δ_i on average. Out of 2000 simulations differing only in their random topology, only 251 did not exceed $\Delta_i > 2$ in at least one metric. Over 1300 even had a deviation of $\Delta_i \geq 100$ in at least one variable, indicating that the trajectory failed to converge. We speculate that this is due to larger separation in phase space of the attractors as well as the larger difference in time scales of the involved oscillations. The widely separated time scales might be better tackled and the prediction errors are alleviated by using different reservoirs adapted to the different time scales. This remains a question for future studies.

Another remaining question concerns the full understanding of the role of noise.[14] In our scheme, noise in the source system helps us to explore all directions in the phase space and enables the system training to succeed. However, noise also results in the destruction of fine details. Finding the best trade-off between noise and accuracy remains a future challenge, which will give important hints as to the kind of information the reservoir is exploiting when inferring the existence of unseen attractors. Conversely, it will be fruitful to study how to achieve improved training on noise-free data or data with minimal noise.

We speculate that the Ridge regression parameter likely plays a crucial role. It controls how much importance the reservoir assigns to small differences in the training data and to what extent it tries to reproduce those. In fact, the Ridge regression parameter might control the "model complexity;" training with strong regularization should lead to simpler models, while training with weak regularization allows for more complexity but is also more susceptible to over-fitting. This also relates to the influence of finite precision and its role in attractor inference.

Taking a broader view, the reservoir has to perform its task based on a finite amount of noisy data. Thus, the amount of information that can be extracted about "unseen parts" of the phase space should also have some fundamental limit. We expect that a more complex target dynamical system will also require a longer training sequence for a reservoir to be able to emulate it. Questions about the relationship of data length, target system complexity, and related fundamental limits will give further insights into how close an "optimal" prediction can be made by a particular reservoir computer.

Furthermore, in a real-world scenario, a way to fully explore the phase space without the need of initial transients from other regions is vital. We suspect that a rough initialization with artificial data such as constant values is possible but have not tested this proposition.

With the examples shown in our work, we provide a proof-of-principle what can be achieved. A reservoir computer can definitely infer the existence of unseen attractors with varying degrees of success. As such, this further proves their suitability as model-free substitutes for a target system.

ACKNOWLEDGMENTS

A.R. and I.F. acknowledge support by the Spanish State Research Agency (AEI) through the Severo Ochoa and Maria de Maeztu Program for Centers and Units of Excellence in R&D (Grant No. MDM-2017-0711).

DATA AVAILABILITY

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

APPENDIX A: CONSTRUCTION OF THE RANDOM ECHO STATE NETWORK

The echo state networks used in this work always consists of $N = 300$ real-valued nodes, whose vectorial state X evolves according to

$$\dot{X} = -X + \tanh\left(W_{\text{res}}X + GW_{\text{in}}u(t) + B\right). \tag{A1}$$

The real-valued reservoir weight matrix W_{res}, input weight matrix W_{in}, and bias vector B are random but fixed and initialized as follows. The bias vector B consists of $N = 300$ uniformly distributed random numbers in $[-b, b]$. W_{in} is a 300×4 dimensional matrix for our system. Each of the $N = 300$ rows has only a single non-zero entry, whose column is randomly chosen with equal probabilities. The real-valued entry itself is drawn from a uniform random distribution in $[-1, 1]$ for each row. The strength of the input weights is globally controlled via parameter G.

Finally, the reservoir weight matrix W_{res} is a sparse matrix with sparsity ρ. We used $\rho = 0.1$ in all presented simulations. Larger ρ were tried but did not yield any immediately noticeable differences, and as sparse matrices are faster to simulate, we chose the smallest acceptable ρ. The non-zero values are initially distributed in $[-1, 1]$. After creation, we calculate the largest real part of the eigenvalues $\text{Re}(\Lambda)_{\text{max}}$ of the resulting matrix W_{sparse}. We then divide the entire matrix and multiply by a margin factor such that the resulting

reservoir weight matrix W_{res} possesses a maximal real part of any eigenvalue with $\text{Re}(\lambda)_{\max}$ that are related through

$$W_{\text{res}} = \text{Re}(\lambda)_{\max} \frac{W_{\text{sparse}}}{\text{Re}(\Lambda)_{\max}}. \tag{A2}$$

In our simulations, $\text{Re}(\lambda)_{\max}$ is either 0.95 or 0.99, depending on what is found to be better.

APPENDIX B: NUMERICAL INTEGRATION OF THE TARGET SYSTEM

The four-dimensional extension of the Lorenz system (4)–(7) as proposed by Li and Sprott[18] is numerically integrated using an Euler–Maruyama integration scheme written in C++. A high-fidelity time series is created with an integration step of $h = 10^{-3}$. From this, we sample the input time series of the reservoir computer every 300 steps for the first parameter regime ($a = 2.0, b = 0.8$), and every 200 steps for the second parameter region ($a = 6.0, b = 0.1$). The parameter h has no connection to θ, which is a reservoir computer parameter, whereas here we are solely concerned with the source time series preparation. This corresponds roughly to taking 15 points per fast oscillation. Slower oscillations are sampled much more, with up to 500 points in the torus or chaotic regions.

To reach the two limit cycles for $a = 2.0$, $b = 0.8$, we start the time series at $(\pm 5, \pm 1, 1 \pm 1)$. To reach the torus solution, we start at $(4, 1, -1, 1)$. To reach the two chaotic attractors for $a = 6.0$, $b = 0.1$, the state is initialized at $(0, \mp 4, 0, \pm 5)$; the torus was reached from $(1, -1, 1, -1)$. We created both trajectories with and without noise. The noise terms ξ_i in (4)–(7) are drawn from a pseudo-random standard Gaussian distribution with mean zero and unit variance. The noise terms in different variables are uncorrelated. For the trajectories with noise, the standard deviation of the noise *in the numerical integration* is set to 0.2; with a time step of $h = 10^{-3}$ this implies that the noise strength σ in (4)–(7) is $0.2/\sqrt{10^{-3}} \approx 6 \times 10^{-3}$. The final sampled time series u_k has length 11 000, of which we use 1000 points for washout, 9999 points for training, and 1 point as reserve to have the future target.

APPENDIX C: RESERVOIR COMPUTING TECHNICAL DETAILS

We integrate (1) with a fourth order Runge–Kutta integrator. The state of all nodes $X_n \in \mathbb{R}$ is initialized as 0 in each element. The integration time step is 0.1. We let the echo state network evolve without input for 300 time units or 3000 steps, and then switch to the "warmup" or "washout" procedure with input. The input sequence u_k is fed as a piece-wise constant function $u(t) \in \mathbb{R}^4$ with interval lengths θ. We use an interval length of $\theta = 2.5$ found through meta-parameter scans, albeit the exact length does not critically influence the performance. Each component of $u(t)$ corresponds to one of the four variables of the Li–Sprott variant of the Lorenz system (4)–(7).

We use the first 1000 points encoded in $u(t)$ to remove any influence of the starting state, i.e., $T_w = 2500$. After that, we start recording the system state for 10^4 inputs, i.e., 25 000 time units, in accordance to (2), i.e., every θ time units the state of all 300 nodes is recorded. We always use the last integration point in each piece-wise constant interval of $u(t)$ to record the maximally large reaction to the input. We obtain a state matrix S with 9999 rows and $N + 1 = 301$ columns.

We create a target vector Y with 9999 rows and four columns. Each column contains one variable of the input u_k shifted one step into the future. Because both input and output are four-dimensional, the output weight matrix W_{out} is of size $(N + 1) \times 4$. We then solve the following equation for W_{out} using the *solve*-function of the *armadillo* C++ wrapper[21] of the Open-BLAS linear algebra package:

$$S^T Y = (S^T S + \eta I_{N+1}) W_{\text{out}}, \tag{C1}$$

where η is the regularization factor of the Tikhonov regularization and I_{N+1} is the identity matrix of appropriate size $N + 1$. S^T is the transpose of the state matrix.

1. Autonomous operation

For the autonomous operation, we use the output weights W_{out} learned during training. First, we reset the system state X to the point it was after the first 300 time units of input-free evolution. We then feed the first 1000 ground-truth points of a noise-free transient u_k leading to one of the three target attractors.

We observe the system state X at the end of this washout-period and construct state vector $S_i(X) \in \mathbb{R}^{301}$ that looks like recording the row of a state matrix (2). But instead of saving its entries, we directly multiply with W_{out}, which yields four values corresponding to the four dimensions of the predicted output stream. We treat these four values as the next element of a self-generating input sequence $\tilde{u}_k$ and keep track of $\tilde{u}_k$. With the new response X generated, we repeat the process for 10 000 steps. The series $\tilde{u}_k$ then is the reconstructed attractor as inferred or learned by the echo state network.

REFERENCES

[1] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," Neural Comput. **14**, 2531–2560 (2002).

[2] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note," Technical Report 148, German National Research Center for Information Technology GMD, Bonn, Germany, 2001.

[3] L. Gonon and J.-P. Ortega, "Reservoir computing universality with stochastic inputs," IEEE Trans. Neural Netw. Learn. Syst. **31**, 100–112 (2020).

[4] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. S. Barbosa, "Next generation reservoir computing," Nat. Commun. **12**, 5564 (2021).

[5] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," Science **304**, 78–80 (2004).

[6] 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).

[7] X. Chen, T. Weng, H. Yang, C. Gu, J. Zhang, and M. Small, "Mapping topological characteristics of dynamical systems into neural networks: A reservoir computing approach," Phys. Rev. E **102**, 033314 (2020).

[8] I. Estébanez, I. Fischer, and M. C. Soriano, "Constructive role of noise for high-quality replication of chaotic attractor dynamics using a hardware-based reservoir computer," Phys. Rev. Appl. **12**, 034058 (2019).

[9] Q. Zhu, H. Ma, and W. Lin, "Detecting unstable periodic orbits based only on time series: When adaptive delayed feedback control meets reservoir computing," Chaos **29**, 093125 (2019).

[10] S. Krishnagopal, G. Katz, M. Girvan, and J. Reggia, "Encoding of a chaotic attractor in a reservoir computer: A directional fiber investigation," in *2019 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2019), pp. 1–8.

[11] J. Z. Kim, Z. Lu, E. Nozari, G. J. Pappas, and D. S. Bassett, "Teaching recurrent neural networks to infer global temporal structure from local examples," Nat. Mach. Intell. **3**, 316–323 (2021).
[12] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," Proc. Natl. Acad. Sci. U.S.A. **113**, 3932–3937 (2016).
[13] T. Qin, K. Wu, and D. Xiu, "Data driven governing equations approximation using deep neural networks," J. Comput. Phys. **395**, 620–635 (2019).
[14] S. A. Billings, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains* (John Wiley & Sons, 2013).
[15] A. Haluszczynski and C. Räth, "Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing," Chaos **29**, 103143 (2019).
[16] A. Haluszczynski, J. Aumeier, J. Herteux, and C. Räth, "Reducing network size and improving prediction stability of reservoir computing," Chaos **30**, 063136 (2020).
[17] Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," Chaos **28**, 061104 (2018).
[18] C. Li and J. C. Sprott, "Coexisting hidden attractors in a 4-D simplified lorenz system," Int. J. Bifurcation Chaos **24**, 1450034 (2014).
[19] Z.-Z. Gao and C. Zhang, "A novel hyperchaotic system," J. Jishou Univ. (Nat. Sci. Ed.) **32**, 65 (2011).
[20] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *2004 IEEE International Joint Conference on Neural Networks, IEEE Cat. No. 04CH37541* (IEEE, 2004), Vol. 2, pp. 985–990.
[21] C. Sanderson and R. Curtin, "Armadillo: A template-based C++ library for linear algebra," J. Open Source Softw. **1**, 26 (2016).