# Automatic-differentiated Physics-Informed Echo State Network (API-ESN) [*]

Alberto Racca[1] and Luca Magri[1,2]

[1] Department of Engineering, University of Cambridge, United Kingdom
[2] (visiting) Institute for Advanced Study, Technical University of Munich, Germany

**Abstract.** We propose the Automatic-differentiated Physics-Informed Echo State Network (API-ESN). The architecture constrains the knowledge of the physical equations through the reservoir's exact time derivative, which is computed by automatic differentiation. As compared to the original Physics-Informed Echo State Network, the accuracy of the time derivative is increased by up to seven orders of magnitude. This increased accuracy is key in chaotic dynamical systems, where errors grows exponentially in time. The architecture is showcased in the reconstruction of unmeasured (hidden) states of a chaotic system. The API-ESN eliminates a source of error, which is present in existing physics-informed echo state networks, in the computation of the time-derivative. This opens up new possibilities for an accurate reconstruction of chaotic dynamical states.

**Keywords:** Reservoir Computing · Automatic Differentiation · Physics-Informed Echo State Network.

## 1 Introduction

In experiments in fluid mechanics, and more broadly in classical physics, we only rarely have measurements on the entire state of the system because of technological/budget constraints on the number and placement of sensors. In fact, we typically measure only a subset of the state, the *observed* states, but we do not have data on the remaining variables, the *hidden* states. In recent years, machine learning techniques have been proposed to infer hidden variables, which is also known as *reconstruction*. A fully-data driven approach to reconstruction assumes that data for the hidden states is available only for a limited time interval, which is used to train the network [8]. On the other hand, a physics-informed approach to reconstruction constrains the governing equations [4,10]. In this work, we use

automatic differentiation to eliminate the source of error in the Physics-Informed Echo State Network, which originates from approximating the time derivative of the network [4]. Automatic differentiation records the elementary operations of the model and evaluates the derivative by applying the chain rule to the derivatives of these operations [2]. With automatic differentiation, we compute exactly the time derivative, thereby extending the network's ability to reconstruct hidden states in chaotic systems. In section 2, we present the proposed architecture: the Automatic-differentiated Physics-Informed Echo State Network (API-ESN). In section 3, we discuss the results. We summarize the work and present future developments in section 4.

## 2    Automatic-differentiated Physics-Informed Echo State Network (API-ESN)

We study the nonlinear dynamical system

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}), \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^{N_y}$ is the state of the physical system, $\mathbf{f}$ is a nonlinear operator, and $\dot{}$ is the time derivative. We consider a case where the state, $\mathbf{y}$, consists of an observed state, $\mathbf{x} \in \mathbb{R}^{N_x}$, and a hidden state, $\mathbf{h} \in \mathbb{R}^{N_h}$: $\mathbf{y} = [\mathbf{x}; \mathbf{h}]$, where $[\cdot\,; \cdot]$ indicates vertical concatenation and $N_y = N_x + N_h$. We assume we have non-noisy data on $\mathbf{x}$, and its derivative, $\dot{\mathbf{x}}$, which can be computed offline. We wish to reconstruct $\mathbf{h}$ given the data. The $N_t$ training data points for the observed states are $\mathbf{x}(t_i)$ for $i = 0, 1, 2, \ldots, N_t$, taken from a time series that ranges from $t_0 = 0$ to $t_{N_t} = N_t \Delta t$, where $\Delta t$ is the constant time step. We introduce the Automatic-differentiated Physics-Informed Echo State Network (API-ESN) to reconstruct $\mathbf{h}$ at the same time instants, by constraining the knowledge of the governing equations through the temporal derivative of the network. The architecture is based on the Physics-Informed Echo State Network (PI-ESN) [3,4], which, in turn, is based on the fully data-driven ESN [5,9]. In the PI-ESN, the time derivative of the network is approximated by a first order forward Euler numerical scheme. In this work, we compute the derivative at machine precision through automatic differentiation [2,1].

In the API-ESN, the data for the observed state is fed as an input and the entire state is predicted as the output. The state of the reservoir at the $i$-th time step, $\mathbf{r}(t_i) \in \mathbb{R}^{N_r}$, is updated as a function of its previous value, $\mathbf{r}(t_{i-1})$, and the current input, $\mathbf{x}(t_i) \in \mathbb{R}^{N_x}$. The output, $\hat{\mathbf{y}}(t_i) = [\hat{\mathbf{x}}(t_{i+1}); \hat{\mathbf{h}}(t_{i+1})] \in \mathbb{R}^{N_y}$, is computed as a linear combination of the components of the state of the reservoir:

$$\mathbf{r}(t_i) = \tanh\left(\mathbf{W}_{\text{in}}[\mathbf{x}(t_i); b_{\text{in}}] + \mathbf{W}\mathbf{r}(t_{i-1})\right); \quad \hat{\mathbf{y}}(t_i) = \mathbf{W}_{\text{out}}[\mathbf{r}(t_i); \mathbf{x}(t_i); 1] \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{N_r \times N_r}$ is the state matrix, $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_r \times (N_x+1)}$ is the input matrix, $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_y \times (N_r+N_x+1)}$ is the output matrix and $b_{\text{in}}$ is the input bias. The input matrix, $\mathbf{W}_{\text{in}}$, and state matrix, $\mathbf{W}$, are sparse, randomly generated and fixed. These are constructed in order for the network to satisfy the echo state

property [9]. The input matrix, $\mathbf{W}_{\text{in}}$, has only one element different from zero per row, which is sampled from a uniform distribution in $[-\sigma_{\text{in}}, \sigma_{\text{in}}]$, where $\sigma_{\text{in}}$ is the input scaling. The state matrix, $\mathbf{W}$, is an Erdős-Renyi matrix with average connectivity $\langle d \rangle$. This means that each neuron (each row of $\mathbf{W}$) has on average only $\langle d \rangle$ connections (non-zero elements). The value of the non-zero elements is obtained by sampling from an uniform distribution in $[-1, 1]$; the entire matrix is then scaled by a multiplication factor to set its spectral radius $\rho$. The only trainable weights are those in the last $N_h$ rows of the output matrix, $\mathbf{W}_{\text{out}}^{(h)}$. The first $N_x$ rows of the output matrix, $\mathbf{W}_{\text{out}}^{(x)}$, are computed through Ridge regression on the available data for the observed state by solving the linear system:

$$\left( \mathbf{R}\mathbf{R}^T + \gamma \mathbf{I} \right) \mathbf{W}_{\text{out}}^{(x)^T} = \mathbf{R}\mathbf{X}^T, \tag{3}$$

where $\mathbf{X} \in \mathbb{R}^{N_x \times N_t}$ and $\mathbf{R} \in \mathbb{R}^{(N_r + N_x + 1) \times N_t}$ are the horizontal concatenation of the observed states, $\mathbf{x}$, and associated reservoir states, $[\mathbf{r}; \mathbf{x}; 1]$, respectively; $\gamma$ is the Tikhonov regularization factor and $\mathbf{I}$ is the identity matrix [9]. $\mathbf{W}_{\text{out}}^{(h)}$ is initialized by solving (3), where we embed prior knowledge of the physics by substituting $\mathbf{X}$ with $\mathbf{H} \in \mathbb{R}^{N_h \times N_t}$, whose rows are constants and equal to the components of the best estimate of the mean of the hidden state, $\overline{\mathbf{h}} \in \mathbb{R}^{N_h}$. The initial output matrix becomes $\mathbf{W}_{\text{out}} = [\mathbf{W}_{\text{out}}^{(x)}; \mathbf{W}_{\text{out}}^{(h)}]$. To train $\mathbf{W}_{\text{out}}^{(h)}$, we minimize the loss function, $\mathcal{L}_{\text{Phys}}$:

$$\mathcal{L}_{\text{Phys}} = \frac{1}{N_t N_y} \sum_{j}^{N_t} ||\dot{\hat{\mathbf{y}}}(t_j) - \mathbf{f}(\hat{\mathbf{y}}(t_j))||^2, \tag{4}$$

where $||\cdot||$ is the $L_2$ norm; $\mathcal{L}_{\text{Phys}}$ is the Mean Square Error between the time derivative of the predicted state, $\dot{\hat{\mathbf{y}}}$, and the right-hand side of the governing equations evaluated at the predicted state, $\hat{\mathbf{y}}$. To compute $\dot{\hat{\mathbf{y}}}$, we need to differentiate $\hat{\mathbf{y}}$ with respect to $\mathbf{x}$, because the time dependence of the system is implicit in the input, $\mathbf{x}(t)$, i.e. $\frac{d\hat{\mathbf{y}}}{dt} = \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{x}} \frac{d\mathbf{x}}{dt}$. The fact that $\dot{\hat{\mathbf{y}}}$ is a function of $\dot{\mathbf{x}}$ means that the accuracy of the derivative of the output is limited by the accuracy of the derivative of the input. In this work, we compute $\dot{\mathbf{x}}$ exactly using the entire state to evaluate $\mathbf{f}(\mathbf{y})$ in (1). Furthermore, $\hat{\mathbf{y}}$ depends on all the $i+1$ inputs up to the current input due to the recurrent connections between the neurons. In Echo State Networks, we have the recursive dependence of the reservoir state, $\mathbf{r}$, with its previous values, i.e., omitting the input bias for brevity

$$\mathbf{r}(t_i) = \tanh \left( \mathbf{W}_{\text{in}}\mathbf{x}(t_i) + \mathbf{W}\tanh \left( \mathbf{W}_{\text{in}}\mathbf{x}(t_{i-1}) + \mathbf{W}\tanh(\mathbf{W}_{\text{in}}\mathbf{x}(t_{i-2}) + \dots) \right) \right). \tag{5}$$

Because of this, the temporal derivative of the current output has to be computed with respect to all the previous inputs in the training set:

$$\dot{\hat{\mathbf{y}}}(t_i) = \frac{d\hat{\mathbf{y}}}{dt}\bigg|_{t_i} = \frac{\partial \hat{\mathbf{y}}(t_i)}{\partial \mathbf{x}(t_i)} \frac{d\mathbf{x}}{dt}\bigg|_{t_i} + \frac{\partial \hat{\mathbf{y}}(t_i)}{\partial \mathbf{x}(t_{i-1})} \frac{d\mathbf{x}}{dt}\bigg|_{t_{i-1}} + \frac{\partial \hat{\mathbf{y}}(t_i)}{\partial \mathbf{x}(t_{i-2})} \frac{d\mathbf{x}}{dt}\bigg|_{t_{i-2}} + \dots \tag{6}$$

This is computationally cumbersome. To circumvent this extra computational cost, we compute $\dot{\hat{\mathbf{y}}}$ through the derivative of the reservoir's state, $\dot{\mathbf{r}} = \frac{\partial \mathbf{r}}{\partial \mathbf{x}} \frac{d\mathbf{x}}{dt}$. By differentiating (2) with respect to time, we obtain $\dot{\hat{\mathbf{y}}} = \mathbf{W}_{\mathrm{out}}[\dot{\mathbf{r}}; \dot{\mathbf{x}}; 0]$. Because $\dot{\mathbf{r}}$ is independent of $\mathbf{W}_{\mathrm{out}}$, $\dot{\mathbf{r}}$ is fixed during training. Hence, the automatic differentiation of the network is performed only once during initialization. Moreover, we can compute $\dot{\mathbf{r}}$ as a function of the current input and previous state as the network evolves:

$$\dot{\mathbf{r}}(t_i) = \frac{d\mathbf{r}}{dt}\bigg|_{t_i} = \frac{\partial \mathbf{r}(t_i)}{\partial \mathbf{x}(t_i)} \frac{d\mathbf{x}}{dt}\bigg|_{t_i} + \frac{\partial \mathbf{r}(t_i)}{\partial \mathbf{r}(t_{i-1})} \frac{d\mathbf{r}}{dt}\bigg|_{t_{i-1}}, \qquad (7)$$

in which we initialize $\dot{\mathbf{r}}(t_0) = 0$ and $\mathbf{r}(t_0) = 0$ at the beginning of the washout interval (the washout interval is the initial transient of the network, during which we feed the inputs without recording the outputs in order for the state of the network to be uniquely defined by the sequence of the inputs [9]).

## 3    Reconstruction of hidden states in a chaotic system

We study the Lorenz system [7], which is a prototypical chaotic system that models Rayleigh–Bénard convection:

$$\dot{\phi}_1 = \sigma_L(\phi_2 - \phi_1), \qquad \dot{\phi}_2 = \phi_1(\rho_L - \phi_3) - \phi_2, \qquad \dot{\phi}_3 = \phi_1\phi_2 - \beta_L\phi_3, \quad (8)$$

where the parameters are $[\sigma_L, \beta_L, \rho_L] = [10, 8/3, 28]$. To obtain the data, we integrate the equation through the implicit adaptive step scheme of the function `odeint` in the `scipy` library. The training set consists of $N_t = 10000$ points with step $\Delta t = 0.01$LTs, where a Lyapunov Time (LT) is the inverse of the leading Lyapunov exponent $\Lambda$ of the system, which, in turn, is the exponential rate at which arbitrarily close trajectories diverge. In the Lorenz system, $\Lambda = \mathrm{LT}^{-1} \simeq 0.906$. We use networks of variable sizes from $N_r = 100$ to $N_r = 1000$, with parameters $\langle d \rangle = 20$, $\gamma = 10^{-6}$ and $\rho = 1.0$ [8]. We set $\sigma_{\mathrm{in}} = 0.1$, $b_{\mathrm{in}} = 10$, and $\overline{h} = 10$, to take into account the order of magnitude of the inputs ($\sim 10^1$). We train the networks using the Adam optimizer [6] with initial learning rate $l = 0.1$, which we decrease during the optimization.

In Fig. 1, we compare the accuracy of the Automatic Differentiation (AD) derivative of the API-ESN with respect to the first-order Forward Euler (FE) approximation of the PI-ESN [4]. Here, we study the case where the entire state is known, $\mathbf{y} = \mathbf{x}$, to be able to compute the true derivative of the predicted state, $\mathbf{f}(\hat{\mathbf{y}})$, (1). In plot (a), we show $\mathbf{f}(\hat{\mathbf{y}})$ in an interval of the training set for $N_r = 100$. In plot (b), we show in the same interval the squared norm of the error for FE, $\mathcal{L}_{\mathrm{FE}}$, and AD, $\mathcal{L}_{\mathrm{AD}}$, with respect to $\mathbf{f}(\hat{\mathbf{y}})$. In addition, we show the squared norm of the error, $\mathcal{L}_{\mathrm{Y}}$, of the output, $\hat{\mathbf{y}}$, with respect to the data, $\mathbf{y}$. Because FE and AD share the same $\hat{\mathbf{y}}$, $\mathcal{L}_{\mathrm{Y}}$ is the same for the two architectures. In plot (c), we show the time average of the squared norms, indicated by the overline, as a function of the size of the reservoir. In the case of $\overline{\mathcal{L}}_{\mathrm{FE}}$ and $\overline{\mathcal{L}}_{\mathrm{AD}}$, they coincide with $\mathcal{L}_{\mathrm{Phys}}$ (4). AD is four to seven orders of magnitude more accurate than FE.
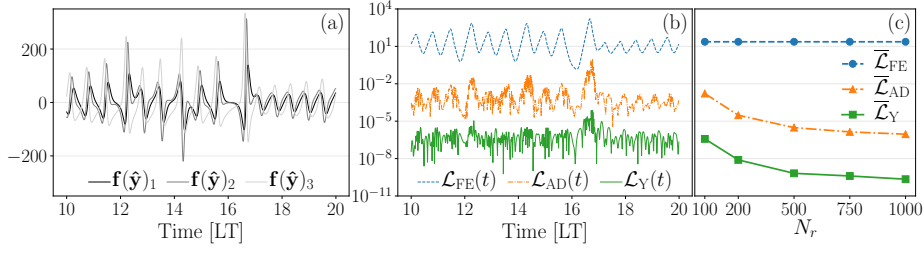
**Fig. 1.** Time derivative of the output for $N_r = 100$ in an interval of the training set, (a). Squared norm of the error of the output derivative, $\mathcal{L}_{\text{FE}}$ and $\mathcal{L}_{\text{AD}}$ , and output, $\mathcal{L}_{\text{Y}}$, in the same interval, (b). Mean Square Error of the output derivative, $\overline{\mathcal{L}}_{\text{FE}}$ and $\overline{\mathcal{L}}_{\text{AD}}$ , and output, $\overline{\mathcal{L}}_{\text{Y}}$, as a function of the reservoir size, (c).

The error of the FE numerical approximation becomes dominant from $N_r = 100$. This prevents its accuracy from increasing in larger networks.

We use the API-ESN to reconstruct the hidden states in three testcases: (i) reconstruction of $\mathbf{h} = [\phi_2]$ given $\mathbf{x} = [\phi_1; \phi_3]$; (ii) reconstruction of $\mathbf{h} = [\phi_3]$ given $\mathbf{x} = [\phi_1; \phi_2]$; (iii) reconstruction of $\mathbf{h} = [\phi_2; \phi_3]$ given $\mathbf{x} = [\phi_1]$. We choose (i) and (ii) to highlight the difference in performance when we reconstruct different variables, and (iii) to compare the reconstruction of the states $\phi_2$ and $\phi_3$ when fewer observed states are available. We reconstruct the hidden states in the training set and in a 10000 points test set subsequent to the training set. The reconstructed states in an interval of the training set for networks of size $N_r = 1000$ are shown in plots (a) and (d) in Fig. 2. The network is able to reconstruct satisfactorily the hidden states. The accuracy deteriorates only in the large amplitude oscillations of $\phi_3$, (d). To visualize the global performance over both the training set and the test set, we plot the Probability Density Functions (PDF) in (b,e), and (c,f), respectively. The PDFs are reconstructed with similar accuracy between the two sets. Interestingly, the increased difficulty in reconstructing $\phi_3$ is due to the dynamical the system's equations (rather than the architecture's ability to learn). Indeed, $\phi_3$ appears in only two of the three equations (8), whereas $\phi_2$ appears in all the equations and it is a linear function in the first equation. In other words, we can extract more information for $\phi_2$ than for $\phi_3$ from the constraint of the physical equations, which means that the network can better reconstruct the dynamics of $\phi_2$ vs. $\phi_3$. In the lower part of large amplitude oscillations of $\phi_3$ in particular, we have small values for the derivatives, so that the error in the governing equations is small.

To quantitatively assess the reconstruction, we compute for each component, $h_i$, of the hidden state, $\mathbf{h}$, the Normalized Root Mean Square Error: $\text{NRMSE}(h_i) = \sqrt{N^{-1} \sum_j^N (\hat{h}_i(t_j) - h_i(t_j))^2} / (\max(h_i) - \min(h_i))$; where $(\max(h_i) - \min(h_i))$ is the range of $h_i$. In Fig. 3, we show the values of the NRMSE for different sizes of the reservoir, $N_r$, in the training (continuous lines) and test (dash-dotted lines) sets for FE, (a,c), and AD, (b,d). The error of the FE approximation dominates the reconstruction of $\phi_2$, (a), while AD produces NRMSEs
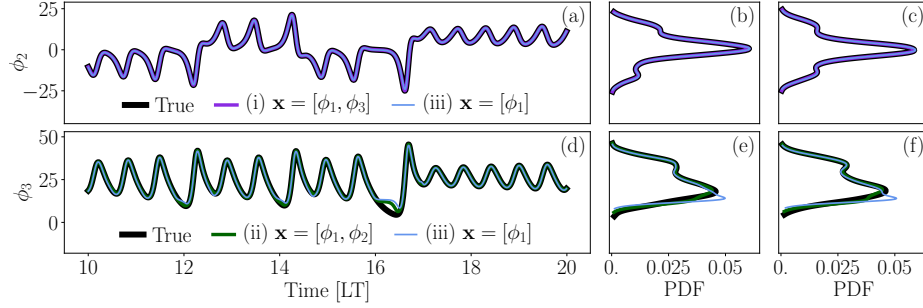
**Fig. 2.** Reconstruction of the hidden variables for $N_r = 1000$ in an interval of the training set, (a) and (d), Probability Density Function (PDF) in the training, (b,e), and test set, (c,f). Reconstruction of $\mathbf{h} = [\phi_2]$ (top row) and $\mathbf{h} = [\phi_3]$ (bottom row): testcase (i) is shown in (a-c), testcase (ii) is shown in (d-f), and testcase (iii) is shown in (a-f).

up to one order of magnitude smaller, (b). In the reconstruction of $\phi_3$, (c) and (d), AD and FE perform similarly because of the mathematical structure of the dynamical system's equations rather than the networks' ability to be trained, as argued for Fig. 2 (e,f). In general, the values of the NRMSE are similar between the training and test sets, which indicates that the reconstruction also works on unseen data. There is a difference between the two sets only in the reconstruction of $\phi_2$ given $\mathbf{x} = [\phi_1; \phi_3]$, (b). In this case, the NRMSE in the test set is smaller than the NRMSE we obtain when using data on the hidden state to train the network, through the architecture described in [8] (results not shown). In addition, we compare the reconstruction of the states when fewer observed states are available. When only $\phi_1$ is available ($\mathbf{x} = [\phi_1]$), the larger error on $\phi_3$, (d), limits the accuracy on the reconstruction on $\phi_2$, (b). This results in a reconstruction of $\phi_3$ with similar accuracy to the case where $\mathbf{x} = [\phi_1; \phi_2]$, while there is a larger error in the reconstruction of $\phi_2$ with respect to the case where $\mathbf{x} = [\phi_1; \phi_3]$.

## 4   Conclusions and future directions

We propose the Automatic-differentiated Physics-Informed Echo State Network (API-ESN) to constrain the knowledge of the governing equations in an echo state network. The exact time-derivative of the dynamical system is computed by automatic differentiation. The time derivative of the output is shown to be a function of all the network's previous inputs because of the recursive time dependence through the neurons' recurrent connections. Albeit this long time-dependence in the past would make the computation of the time derivative computationally cumbersome, we eliminate this cost by computing the derivative of the reservoir's state. We apply the API-ESN to a prototypical chaotic system to reconstruct the hidden states in different datasets both in the training points
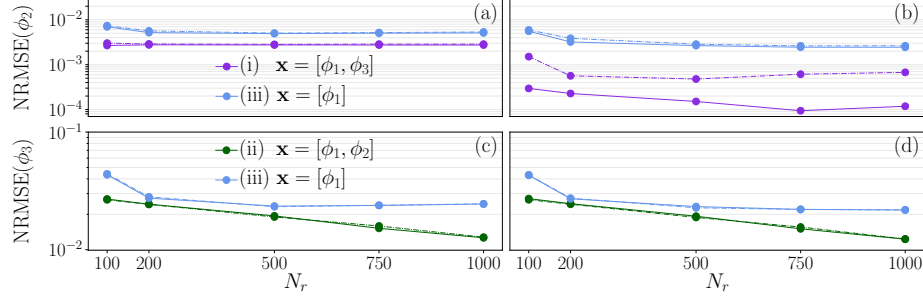
**Fig. 3.** NRMSE for in the training (continuous lines) and test (dash-dotted lines) sets as a function of the size of the reservoir. The reconstruction is performed through forward Euler, (a,c), and automatic differentiation, (b,d). Reconstruction of $\mathbf{h} = [\phi_2]$, (top row) and $\mathbf{h} = [\phi_3]$ (bottom row): testcase (i) is shown in (a,b), testcase (ii) is shown in (c,d), and testcase (iii) is shown in (a-d).

and on unseen data. We compare the API-ESN to the forward Euler approximation of the Physics-Informed Echo State Network. We obtain a Normalized Mean Square Error up to one order of magnitude smaller in the reconstruction of the hidden state. Current work is focused on optimizing the API-ESN to reconstruct hidden states in closed-loop prediction.

## References

1. Abadi, M., et al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/
2. Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in machine learning: a survey. The Journal of Machine Learning Research **18**(1), 5595–5637 (2017)
3. Doan, N.A.K., Polifke, W., Magri, L.: Physics-informed echo state networks for chaotic systems forecasting. In: ICCS. pp. 192–198. Springer (2019)
4. Doan, N.A.K., Polifke, W., Magri, L.: Learning hidden states in a chaotic system: A physics-informed echo state network approach. In: ICCS. pp. 117–123. Springer (2020)
5. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
7. Lorenz, E.N.: Deterministic nonperiodic flow. J. Atmos. Sci. **20**(2), 130–141 (1963)
8. Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R., Ott, E.: Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. Chaos: An Interdisciplinary Journal of Nonlinear Science **27**(4), 041102 (2017)
9. Lukoševičius, M.: A practical guide to applying echo state networks. In: Neural networks: Tricks of the trade, pp. 659–686. Springer (2012)
10. Raissi, M., Yazdani, A., Karniadakis, G.E.: Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science **367**(6481), 1026–1030 (2020)